LEIBNIZ UNIVERSITÄT HANNOVER
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK
INSTITUT FÜR PRAKTISCHE INFORMATIK
FACHGEBIET DATENBANKEN UND INFORMATIONSSYSTEME

Masterarbeit

im Studiengang Informatik

Analyse und Erweiterung von Methoden des Data Mining in räumlichen Datenbanken

Markus Spehling
Matrikel.-Nr. 2036328

Prüfer: Prof. Dr. Udo Lipeck
Zweitprüfer: Dr. Hans Hermann Brüggemann
Betreuer: Dipl.-Math. Christian Stahlhut

30. März 2007

Zusammenfassung

In den vergangenen Jahrzehnten ist – durch die Möglichkeit der Speicherung von großen Datenmengen – das Wachstum an gespeicherten und zu analysierenden Daten rasant angestiegen. Das Data Mining beschreibt dabei eine Technik zur automatisierten Analyse. In vielen Bereichen des täglichen Lebens wird diese Möglichkeit der Analyse genutzt, wobei dabei auf eine Vielzahl von Implementierungen zurückgegriffen wird.

In räumlichen Datenbanken ist das Data Mining jedoch aufgrund besonderer Eigenschaften von räumlichen Daten nicht mit den *herkömmlichen* Techniken realisierbar, weshalb die vorhandenen Implementierungen der Data Mining-Methoden nicht genutzt werden können.

In dieser Arbeit wird ein Konzept zur Materialisierung von räumlichen Informationen entwickelt, wodurch das Data Mining in räumlichen Datenbanken ermöglicht werden soll, indem ausschließlich Methoden und Algorithmen für das Data Mining in relationalen Datenbanken benutzt werden. Dabei richtet sich das Vorgehen der Entwicklung des Konzepts an die in der Literatur diskutierten Problemstellungen und Lösungsvorschläge zur Materialisierung für das Data Mining in räumlichen Datenbanken.

Für eine geeignete Implementierung eines Data Mining-Systems, welches Methoden und Algorithmen für das Data Mining in relationalen Datenbanken bereitstellt, fällt die Entscheidung auf die Data Mining-Cartridge von Oracle. Sowohl der Funktionsumfang als auch die Funktionsweise werden dabei grundlegend analysiert, wobei ebenfalls die Einsatzfähigkeit und die Grenzen des Data Mining-Systems für das Data Mining in relationalen Datenbanken überprüft wird. Die Verwendung der Data Mining-Cartridge wird im Anschluss daran anhand von Beispielen veranschaulicht.

Abschließend wird die Implementierung des Konzepts zur Materialisierung und die Integration in das Datenbank-Management-System von Oracle vorgestellt. Das Data Mining in räumlichen Datenbanken wird dann durch die Kombination dieser Erweiterung, in Form einer Vorverarbeitung, mit der Oracle Data Mining-Cartridge demonstriert.

Inhaltsverzeichnis

1 Einleitung **6**

2 Knowledge Discovery und Data Mining **8**
2.1 Übersicht zum KDD-Prozess . . . 8
2.2 Erfassung von Hintergrundwissen und Zielen . . . 9
2.3 Selektion . . . 9
2.4 Preprocessing . . . 10
2.4.1 Behandlung fehlender Werte (Missing Values) . . . 10
2.4.2 Behandlung von Ausreißern (Outlier) . . . 11
2.4.3 Behandlung von Redundanzen . . . 11
2.4.4 Behandlung von Inkonsistenzen . . . 12
2.5 Transformation . . . 12
2.5.1 Normierung . . . 12
2.5.2 Diskretisierung . . . 13
2.5.3 Aggregierung . . . 14
2.5.4 Generalisierung . . . 14
2.5.5 Attribut-Konstruktion . . . 14
2.5.6 Attribut-Reduktion . . . 14
2.6 Data Mining . . . 14
2.6.1 Übersicht zum Data Mining . . . 15
2.6.2 Clustering . . . 16
2.6.3 Assoziationsanalyse . . . 22
2.6.4 Klassifikation . . . 26
2.6.5 Regression . . . 33
2.7 Interpretation . . . 35

3 Oracle Knowledge Discovery und Data Mining **36**
3.1 Überblick . . . 36
3.2 Programmable Interface . . . 37
3.2.1 PL/SQL-Schnittstelle . . . 37
3.2.2 Java-Schnittstelle . . . 38
3.3 Voraussetzungen . . . 38
3.4 Management-Prinzip von Oracle Data Mining . . . 39
3.5 Oracle Knowledge Discovery . . . 41
3.5.1 Oracle Preprocessing . . . 41
3.5.2 Oracle Transformation . . . 46
3.5.3 Oracle Data Mining . . . 51

4 Data Mining - Ein Beispiel **73**
4.1 Assoziationsanalyse . . . 74
4.1.1 Assoziationsanalyse: `Schauspieler mit Schauspieler in Filmen` . . . 74
4.1.2 Erfahrungen mit der Assoziationsanalyse in ODM . . . 78

4.2 Regression . . . 79
4.2.1 Regressionsanalyse: `Jahr und Anzahl produzierter Filme pro Jahr` . . . 79
4.2.2 Erfahrungen mit der Regressionsanalyse in ODM . . . 83
4.3 Clustering . . . 83
4.3.1 Clustering: `Demographische Analyse der Schauspieler` . . . 83
4.3.2 Erfahrungen mit der Clusteranalyse in ODM . . . 87
4.4 Klassifikation . . . 88
4.4.1 Klassifikation: `Einfluss von Schauspielern auf die Bewertung des Films` 88
4.4.2 Erfahrungen mit der Klassifikation in ODM . . . 92

5 Data Mining in räumlichen Datenbanken 93
5.1 Räumliche Datenbanken . . . 93
5.2 Spatial Data Mining . . . 94
5.2.1 Problemstellungen des Spatial Data Mining . . . 96
5.2.2 Lösungsansätze für das Spatial Data Mining . . . 99
5.2.3 Frameworks für Spatial Data Mining . . . 105
5.2.4 Zusammenfassung . . . 109
5.3 Materialisierung von geometrischen Objekten und räumlichen Beziehungen . . . 110
5.3.1 Beispiel für die Materialisierung von räumlichen Informationen . . . 110
5.3.2 Was wird für das Spatial Data Mining materialisiert? . . . 111
5.3.3 Transformation geometrischer Objekte . . . 112
5.3.4 Materialisierung von Nachbarschaftsbeziehungen . . . 119
5.3.5 Materialisierung von topologischen Beziehungen . . . 120
5.3.6 Materialisierung von metrischen Beziehungen . . . 121
5.3.7 Materialisierung von gerichteten Beziehungen . . . 123

6 Implementierung 124
6.1 Spatial Data Mining-Transformation . . . 125
6.1.1 Materialisierung von räumlichen Beziehungen . . . 125
6.1.2 Materialisierung von geometrischen Objekten . . . 128
6.2 Nutzen von Materialisierungen für das Spatial Data Mining . . . 129
6.2.1 Fallbeispiel: Clustering von Ballungsgebieten . . . 130
6.2.2 Fallbeispiel: Beziehungen zwischen Geo-Daten . . . 134

7 Ausblick 137

A Spatial Data Mining - Beispiele 139
A.1 Clustering von Ballungsgebieten . . . 139

B Oracle Data Mining - Beispiele 141
B.1 Assoziationsanalyse . . . 141
B.2 Regressionsanalyse . . . 142
B.3 Clustering . . . 144
B.4 Klassifikation . . . 145

Literaturverzeichnis 148

1 Einleitung

In den vergangenen Jahrzehnten kam es in ziemlich jedem Bereich unseres Lebens zu einem enormen Wachstum an gesammelten Daten, die in immer größer werdenden Datenbanken oder Datenbank-Clustern gespeichert sind. Dazu zählen z. B. das Internet, die Telekommunikationsbranche (Verbindungsdaten, etc.) und Geographische Informationssysteme (Karten- und Standortinformationen)[Atk06], um nur einige wenige zu nennen. Dieser rasante Anstieg an gespeicherten Daten hat jedoch zur Folge, dass die Fähigkeit der menschlichen Analyse bei weitem überschritten ist. Die Konsequenz sind Datengräber (*data tombs*) [HK00], d. h. Archive, die überwiegend Daten speichern, aber nur in geringem Maße wieder angefragt oder ausgewertet werden. Die gespeicherten Daten können jedoch möglicherweise interessante Informationen enthalten, die nicht explizit gespeichert sind und somit auch nicht durch die Anfragenmöglichkeiten *normaler* Datenbanksysteme oder durch maschinell-erstellte Reports (Reportgeneratoren) ermittelt werden können. Es lassen sich somit zwar beliebige Einzelinformationen oder auch Aggregierungen von Informationen ermitteln, jedoch Muster, Strukturen und Regelmäßigkeiten, die allgemein als Gesetzmäßigkeiten bezeichnet werden, bleiben dabei unbemerkt. Aus diesem Grund wurde der Begriff des Data Mining, also das Schürfen nach Daten in Datenmengen, zum Aufdecken von impliziten Informationen geprägt. Darüber hinaus wurde der komplette Prozess um das Data Mining als Knowledge Discovery in Databases (KDD) bezeichnet, in dem es einen wichtigen Bestandteil repräsentiert.

Data Mining ist ein Forschungsbereich in der Informatik, der sich seit den späten 80iger Jahren des 20. Jahrhunderts bis heute stark weiterentwickelt hat. In dieser Zeit sind Verfahren entwickelt worden, die die Suche nach versteckten und interessanten Informationen in Daten ermöglichen, um daraus Schlussfolgerungen über Gesetzmäßigkeiten ziehen zu können, ohne vorher zu wissen, welcher Art diese sind. Die Gesetzmäßigkeiten werden dem Benutzer im Anschluss an den Data Mining-Prozess in einer verständlichen Form präsentiert. Das Data Mining findet dabei in vielen verschiedenen Bereichen Anwendung, z. B. in Texten (Text-Mining), im Internet (Web-Mining), in Bildern und Filmen (Multimedia-Mining) und in Daten in Form von Tabellen.

Die Datenbanksystem-Entwickler (Oracle, Microsoft, IBM, etc .) haben die enorme Bedeutung des Data Mining und den steigenden Einsatz und Nutzen in jüngster Zeit durch Unternehmen erkannt und haben ihre Produkte um diese Errungenschaften erweitert, so dass das Data Mining komfortabler durchgeführt werden kann. Oracle, beispielsweise, stellt für das Data Mining in relationalen Datenbanken die Data Mining-Cartridge zur Verfügung. Darin sind dafür einige gängige und gut erforschte Algorithmen für das Data Mining und allgemein für den KDD-Prozess integriert.

In dieser Arbeit soll das Potential von Oracle in Bezug auf Data Mining in räumlichen Datenbanken untersucht und erweitert werden. Die Hauptbestandteile der Untersuchung werden die Data Mining-Cartridge, die Spatial-Cartridge und die Verbindung zwischen den beiden Cartridges von Oracle sein. Die grobe Vorgehensweise ist so konzipiert, dass zunächst allgemein der KDD-Prozess und das Data Mining und anschließend die Spezialisierung im Fall des räumlichen Data Mining beschrieben wird. Die Arbeit ist, wie nachfolgend beschrieben, unterteilt.

Das zweite Kapitel beschäftigt sich mit der Knowledge Discovery und dem Data Mining. Dabei wird auf den KDD-Prozess eingegangen: welche Schritte beinhaltet er und wie ordnet sich das Data Mining in den Prozess ein. Außerdem sollen u. a. folgende Fragen geklärt werden:

(1) Welche konkreten Ziele werden mit dem Data Mining verfolgt?
(2) Welche Data Mining-Verfahren gibt es?
(3) Wie werden die Data Mining-Verfahren eingesetzt?

Im dritten Kapitel soll die Knowledge Discovery und das Data Mining in Oracle untersucht werden. Dabei werden die Data Mining-Verfahren (Methoden), die im Oracle-DBMS (Datenbank-Management-System) integriert sind, genauer analysiert und beschrieben. Hierbei sollen die Stärken, Schwächen und Beschränkungen des Oracle Data-Mining-Systems hinsichtlich der im zweiten Kapitel vorgestellten Grundlagen untersucht werden. Außerdem soll das Konzept, das hinter der Data Mining-Cartridge steckt, beleuchtet werden.

Anschließend wird im vierten Kapitel die Anwendung des Data Mining, unter Benutzung der in Kapitel 3 vorgestellten Methoden, beispielhaft an Problemstellungen des Data Mining durchgeführt.

Im fünften Kapitel soll die Anwendung des Data Mining auf räumliche Datenbanken und die Unterstützung von Oracle dahingehend untersucht werden. Es soll ebenso auf Forschungsarbeiten zur Optimierung und neue Verfahren im Bereich der räumlichen Datenbanken eingegangen werden. Dabei wird außerdem untersucht, welche zusätzlichen Anforderungen der räumliche Aspekt bewirkt und welche Erkenntnisse bzw. Neuerungen durch das räumliche Data Mining erzielt wird. Des Weiteren wird ein Konzept vorgestellt, mit dem räumliches Data Mining durch die Materialisierung räumlicher Aspekte mit der Data Mining-Cartridge von Oracle durchgeführt werden kann.

Im sechsten Kapitel, dem praktischen Teil der Arbeit, wird das in Kapitel 5 vorgestellte Konzept implementiert und in der Datenbank-Management-System von Oracle integriert. Durch Beispiele wird der Umgang und der Nutzen der Implementierung für das Data Mining in räumlichen Datenbanken demonstriert.

2 Knowledge Discovery und Data Mining

In diesem Kapitel wird die Thematik der Knowledge Discovery in Databases (KDD) und des Data Mining, das eng mit dem KDD in Beziehung steht, beschrieben. Die KDD ist eine Technik, auch als KDD-Prozess bezeichnet, zur Datenanalyse, die immer dann Anwendung findet, wenn in den Daten unbekannte bzw. verborgene Beziehungen oder Korrelationen vermutet werden. Der KDD-Prozess und das Data Mining werden also eingesetzt, wenn die herkömmlichen Datenanalyseverfahren nicht mehr praktikabel oder ausreichend sind, d. h. die zu analysierende Datenmenge nicht mehr überschaubar und die Fähigkeit der menschlichen Analyse überschritten ist oder die Gefahr droht, dass ein Großteil der Daten nicht analysiert wird. Die bisherigen Methoden zur Analyse von Daten leiden außerdem unter dem Aspekt recht statisch zu sein (Reportgeneratoren und SQL-Anfragen), oberflächlich zu arbeiten und eher einen Überblick oder eine einfache Auswertung der Daten wiederzugeben. Diese Methoden liefern häufig nur Informationen, die zum einen bereits bekannt sind und zum anderen nur eine Teilmenge der *interessanten* Daten repräsentieren. Es wird jedoch nicht versucht, die Daten tiefgründiger zu analysieren und „über den Tellerrand zu blicken“, um nicht-offensichtliche und verborgene Beziehungen zu finden. Das ist der Punkt, an dem der KDD-Prozess und das Data Mining ansetzen. Durch das Data Mining sollen interessante – bisher unbekannte – Informationen in Form von Gesetzmäßigkeiten automatisch gefunden werden, die jedoch nicht explizit gespeichert sind.

2.1 Übersicht zum KDD-Prozess

Die Knowledge Discovery lässt sich als nicht-trivialer Prozess zur Identifikation von gültigen, neuartigen, potentiell nützlichen und allgemein verständlichen Mustern in Daten beschreiben (*„Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potential useful, and ultimately understandable pattern in data“*)[FSM92]. Die allgemeine Intention der Suche nach verborgenen Informationen in großen Datenmengen, die der KDD-Prozess zweifellos verfolgt, ist spannend und reizvoll zugleich, jedoch stellt sich dieser Prozess im Allgemeinen als recht schwierig dar. Es kommt beispielsweise die Frage auf, wann ein Muster *neuartig* ist oder wann ein Muster als *potentiell nützlich* erachtet wird, und wer das alles letztendlich entscheidet.

Im Idealfall läuft der Prozess der Knowledge Discovery vollautomatisch und geradlinig ab. In der Praxis wird jedoch eher beobachtet, dass der vollautomatische, lineare Prozess eher ein interaktiver und iterativer Prozess ist, bei dem mehrere Schleifen enthalten sein können und bei dem der Anwender häufig korrigierend eingreifen muss, da sonst falsche und unsinnige Ergebnisse oder wo möglich gar keine Ergebnisse aus dem KDD-Prozess hervorgehen.

In Abb. 2.1 wird dieser interaktive und iterative Charakter des Prozesses übersichtlich dargestellt. Der KDD-Prozess unterteilt sich danach in die nachfolgenden sechs Schritte [FPSS96] [HK00], wobei der erste Schritt nicht direkt dargestellt ist, da dieser die Auswahl der Datenbank (z. B. Bibliotheksdatenbank oder Studentendatenbank) betrifft.

1. Erfassung von Hintergrundwissen und Zielen.
2. Selektion - Datenauswahl und Datenintegration.
3. Preprocessing - Datenvorverarbeitung, Datenbereinigung.

4. Transformation - Datenreduktion und Datentransformation.
5. Data Mining - die eigentliche Datenanalyse.
6. Interpretation - Auswertung der gewonnenen Erkenntnisse.

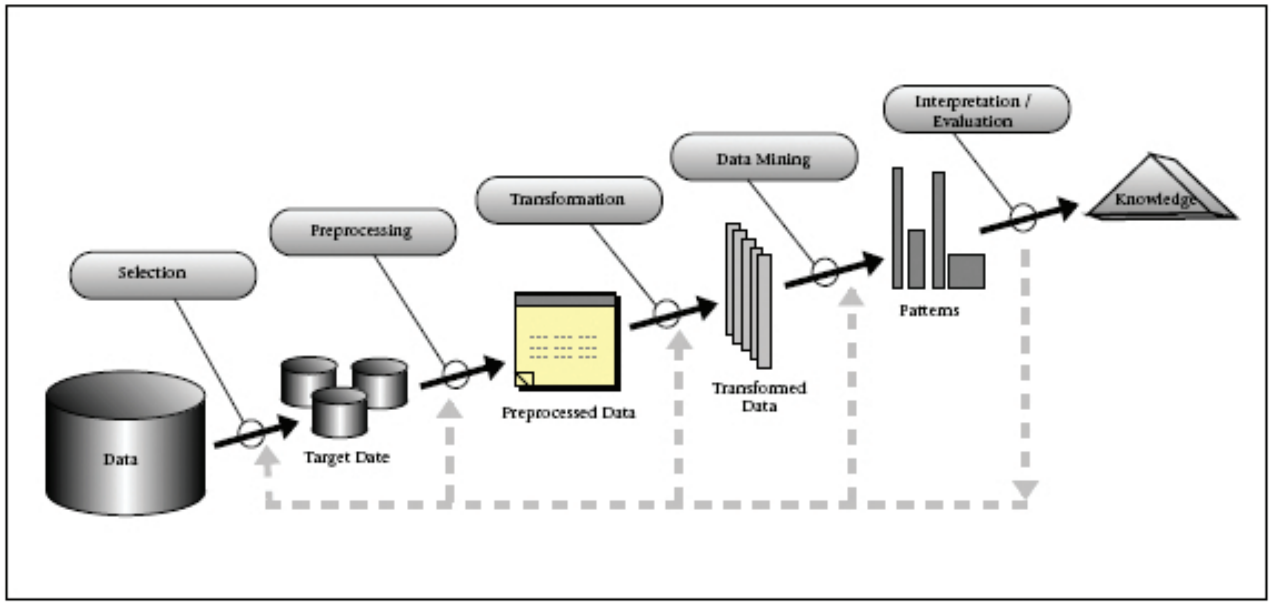


Abbildung 2.1: Überblick über die Stufen des KDD Prozesses[FPSS96]

Der KDD-Prozess ist so aufgebaut, das der 5. Schritt, das Data Mining, eine zentrale Rolle einnimmt, in dem die eigentliche Analyse, d. h. die Suche nach verborgenen Informationen, stattfindet. Die Schritte 1-4 vor dem Data Mining haben die wichtige Aufgabe die Daten so bereitzustellen, dass die Data Mining-Verfahren brauchbare und verwertbare Ergebnisse liefern können, so dass im 6. Schritt die gefundenen Ergebnisse ausgewertet und interpretiert werden können. Der interaktive und iterative Charakter ergibt sich aus dem Aspekt, dass keine zufriedenstellenden Ergebnisse in Bezug auf ein gesetztes Ziel erzielt worden sind, und dadurch Anpassungen in den Schritten 1-4 oder Änderungen an den Einstellungen der Data Mining-Verfahren vorgenommen werden müssen. Diese Anpassungen werden so lange durchgeführt, bis das gewünschte Ergebnis erzielt ist oder bis die Erkenntnis da ist, dass keine signifikanten verborgenen Informationen enthalten sind. Das restliche Kapitel befasst sich mit einer detaillierteren Beschreibung der einzelnen Schritte des KDD-Prozesses.

2.2 Erfassung von Hintergrundwissen und Zielen

In diesem Schritt werden die Datengrundlage und die angestrebten Ziele des Anwenders spezifiziert. Dazu wird zunächst ein Verständnis für das Arbeitsgebiet (Wissensgebiet) entwickelt, wobei ebenfalls existierendes und vorhandenes Wissen mit einbezogen wird. Durch die Zielsetzung des Anwenders wird bereits eine Vorselektion der anwendbaren Data Mining-Verfahren vorgenommen.

2.3 Selektion

In diesem Schritt wird eine Datenmenge definiert, auf der die Knowledge Discovery durchgeführt werden soll. Im Fokus des Interesses stehen Untermengen von verfügbaren Attributen oder Ausschnitte aus der Datenmenge, die genauer betrachtet werden sollen. Es werden Daten, die nicht in

den weiteren Prozess mit einbezogen werden sollen, in diesem Schritt eliminiert. Dies ist ähnlich zum Data Warehousing, wo ebenfalls eine Auswahl der Daten vorgenommen wird, die zu *Data Cubes*[1] zusammengefasst werden und auch nur eine eingeschränkte Sicht auf die gesamten Informationen bieten. Der Zugriff auf diese *Data Cubes* erfolgt über OLAP-Methoden[2].

Bei der Auswahl der Daten, die für das Data Mining verwendet werden sollen, spielt die Datenintegration eine wichtige Rolle [Len02]. Wenn beispielsweise aus verschiedenen Quellen Informationen benötigt werden, müssen diese zu einer einzigen zusammengefasst werden. Probleme, die bei der Datenintegration auftreten können, sind im Folgenden aufgeführt [HK00]:

- Schema-Integration und das damit verbundene Entity-Identification-Problem [Lip04]
- Redundanz in den Daten
- Entdecken und Auflösen von Konflikten in den Daten

Die Selektion und die damit verbundene Datenintegration ist essentiell, da die nachfolgenden Schritte des KDD-Prozesses und insbesondere die Methoden im Data Mining-Schritt nicht mit mehreren Quellen arbeiten können.

2.4 Preprocessing

Im Preprocessing-Schritt sollen die Daten aufgearbeitet und von Fehlern bereinigt werden. Ein gravierendes und häufig vorkommendes Problem heutiger Datensammlungen, die in relationalen Datenbanksystemen gespeichert sind, liegt darin, dass die Datensammlungen unvollständige Daten (d. h. Attributwerte fehlen oder nur aggregierte Attributwerte vorhanden sind), störende Daten (Fehler oder Ausreißer), inkonsistente Daten und Redundanzen aufweisen. Darum ist es im Preprocessing-Schritt notwendig, dass diese Unstimmigkeiten behoben und für den Data Mining-Schritt vorbereitet werden. Dadurch soll erreicht werden, dass die Data Mining-Verfahren schnell und effizient arbeiten können und das Ergebnis qualitativ besser und die gefundenen Gesetzmäßigkeiten aussagekräftiger sind. Zum Preprocessing gehören nachfolgende Teilschritte:

2.4.1 Behandlung fehlender Werte (Missing Values)

Das Problem der fehlenden Werte tritt in relationalen Datenbanken relativ häufig auf, da oftmals dieselbe Anzahl an Attributen für jeden Datensatz auch dann gefordert ist, wenn für einige Attribute keine Eintragungen möglich sind oder unwichtig erscheinen. Die fehlenden Werte stören hingegen den eigentlichen Betrieb der Datenbank nicht, sind jedoch erheblich für das Data Mining. Zum Beispiel stehen in Fragebögen eine Menge von Feldern zum Ausfüllen bereit, wobei einige obligatorisch und andere optional sind. Es muss weiterhin zwischen Fehlwerten, die gewollt sind (optionale Felder im Fragebogen), und Fehlwerten, die nicht angegeben wurden (obligatorische Felder im Fragebogen) unterschieden werden. Ursachen für ersteres könnte sein, dass einige Felder nicht als so wichtig erachtet und darum nur sporadisch angegeben werden und Ursachen für zweiteres könnte Nachlässigkeit beim Ausfüllen des Fragebogens sein.

Ein weiteres Beispiel ist der *Warenkorb*, bei dem der Datensatz dem Warenkorb entspricht und die Attribute den Waren entsprechen. In diesem Fall sind fehlende Werte normal und müssen nicht behandelt werden, da davon ausgegangen werden kann, dass lediglich eine kleine Teilmenge des Warenangebotes gekauft wird.

Zur Behandlung der fehlenden Werte stehen einige Techniken zur Verfügung, die die fehlenden Werte *sinnvoll* auffüllen [HK00][WIK06][DLR77]:

[1] Data-Cube: Speicherung von mehrdimensionalen aggregierten Daten. Einsatz bei OLAP und Data Warehousing.
[2] OLAP = Online Analytical Processing

1. Datensatz mit fehlenden Werten ignorieren
2. Manuelles Auffüllen der fehlenden Werte
3. Fehlende Werte mit einer globalen Konstante auffüllen, z. B. *Unknown*.
4. Mittelwert zum Auffüllen verwenden
5. Wahrscheinlichsten Wert zum Auffüllen verwenden [DLR77]

2.4.2 Behandlung von Ausreißern (Outlier)

Bei Ausreißern handelt es sich häufig um Daten, die vom Standard oder vom Erwarteten abweichen, z. B. eine Person, die ein Studium im Alter von 16 Jahren beginnt. [Haw80] beschreibt einen Ausreißer wie folgt: „ Ein Ausreißer ist eine Beobachtung, die so weit von anderen Beobachtungen abweicht, so dass der Verdacht entsteht, dass er durch einen anderen Mechanismus generiert wurde.“ Es bleibt offen, ob es sich um fehlerhafte oder um korrekte Daten handelt, die interessante Schlussfolgerungen zulassen oder seltene Sondersituationen darstellen?[3] Zur Identifizierung und Eliminierung von Ausreißern stehen einige Methoden [KN97][NMV92][Haw80] zur Verfügung, wobei bei allen die Grundidee gleich ist: Eliminierung der Ausreißer durch Anpassung der Werte an ihre Umgebung [Kie06][HK00]. Es folgt eine Auflistung von einigen Methoden, die zur Identifikation und Eliminierung von Ausreißern verwendet werden:

1. **Diskretisierung:** Die Daten sortieren und in Partitionen unterteilen, und anschließend die Daten durch, beispielsweise Durchschnitt, Median oder Grenzen der Partitionen glätten.
2. **Clipping:** Beim Clipping werden einfach die obersten und die untersten Werte, also die extremen Werte, besonders behandelt (z. B. 10% der Werte, also 5% oberste und 5% unterste Werte). Dabei werden zwei Strategien verfolgt, zum einen das **Winsorizing**, das die obersten und untersten Werte durch spezifizierte Werte ersetzt und zum anderen das **Trimming**, das die obersten und untersten Werte einfach entfernt und somit ignoriert.
   Beispiel: Es seien folgende Daten gegeben: $1, 5, 7, 8, 9, 10, 18$. Beim Trimming werden die extremen Werte einfach ignoriert und die Menge der Werte sinkt auf: $5, 7, 8, 9, 10$. Beim Winsorizing werden die extremen Werte durch vorgegebene Werte ersetzt und die Menge der Werte bleibt konstant: $5, 5, 7, 8, 9, 10, 10$.
3. **Clustering:** Das Clustering, das eigentlich eine Data Mining-Methode ist, um Daten anhand bestimmter Kriterien zu gruppieren (siehe auch 2.6.2), kann dazu benutzt werden, um Ausreißer zu identifizieren und zu eliminieren.
4. **Semi-automatische Analyse:** Die Ausreißer werden durch ein beliebiges Verfahren identifiziert, und danach durch manuelle Überprüfung entfernt oder nicht.
5. **Regression:** Die Regression, die ebenfalls eine Data Mining-Methode ist, um numerische Werte mittels einer Regressionsfunktion zu bestimmen bzw. vorherzusagen (siehe auch 2.6.5), kann zur Identifizierung und Eliminierung benutzt werden.

2.4.3 Behandlung von Redundanzen

Redundanzen entstehen, wenn Namenskonventionen und Normierungen in den Datenmodellen fehlen. Durch fehlende Namenskonventionen kann es vorkommen, dass gleiche Attribute in verschiedenen Quellen unterschiedliche Bezeichnungen besitzen, und bei der Zusammenführung der Daten, semantisch betrachtet, doppelt auftreten und implizit eine Abhängigkeit suggerieren, z. B. ein Attribut `A` lässt sich durch eine Abbildung `f` in Attribut `B` überführen, wodurch das Attribut `B` redundant wird, da die Attribute `A` und `B` das gleiche beschreiben.

[3] One person’s noise is another person’s signal.[War04]

Fehlende Normalisierungen der Datenmodelle verursachen ähnliche Probleme wie die fehlenden Namenskonventionen, wobei die Ursache in der Verwendung unterschiedlicher Wertebereiche der Attribute liegt, wodurch funktionale Abhängigkeiten zwischen Attributen auftreten können. Zur Behandlung von Redundanzen sind daher Namenskonventionen und Normalisierungen der Datenmodelle notwendig. Die Namenskonventionen müssen hierbei manuell eingeführt werden, während die Normalisierungen weitestgehend automatisch durchgeführt werden können. Ein Verfahren zum Finden von Redundanzen von Attributen ist die Korrelationsanalyse.

2.4.4 Behandlung von Inkonsistenzen

Bei Inkonsistenzen handelt es sich um Unstimmigkeiten in den Daten, z. B. ein Kind hat einen Job und ein geregeltes Einkommen Die Unstimmigkeiten sind i. d. R. recht einfach zu identifizieren, jedoch schwierig zu beheben, da sie Hintergrundwissen voraussetzen. Das Kind ist in Wirklichkeit z. B. ein Erwachsener, d. h. die Altersangabe ist fehlerhaft vs. es handelt sich um ein Kind und die Informationen über den Job und das Einkommen sind fehlerhaft. Inkonsistenzen können durch einen Vergleich mit externen Referenzen oder unter Berücksichtigung von Integritätsbedingungen identifiziert und entgegengewirkt werden. Eine automatisierte Behandlung von bestehenden Inkonsistenzen ist aus den genannten Aspekten nicht durchführbar.

2.5 Transformation

Im Transformationsschritt werden die vorverarbeiteten Daten aus dem Preprocessing-Schritt in eine Form transformiert oder zusammengefasst, die für die Data Mining-Methoden zweckdienlich und geeignet sind. Durch die Transformation soll eine Abstraktion der Daten und Struktur- und Formatvereinheitlichung der Daten erreicht werden, so dass eine weitere Verarbeitung im Hinblick auf das eigentliche Data Mining sichergestellt werden kann. Eine Transformation der Daten kann mit den nachfolgenden Methoden erzielt werden [HK00]:

2.5.1 Normierung

Bei der Normierung wird der Wertebereich durch eine Normierungsfunktion auf einen definierten Bereich abgebildet. Dadurch wird eine Vergleichbarkeit von Werten erreicht, die zuvor nicht möglich war aufgrund unterschiedlicher Wertebereiche, z. B. Einkommen von Angestellten in verschiedenen Branchen. Die Normierung beschränkt sich auf numerische Werte. Folgende Normierungstechniken (mit Normierungsfunktion) werden eingesetzt:

1. **Min-Max-Normierung:** $v'(v) = \frac{v-min_A}{max_A-min_A}(new\_max_A - new\_min_A) + new\_min_A$ mit $v \in dom(A)$ und $v' \in [new\_min_A, new\_max_A]$. Dabei handelt es sich bei v um den Wert des Attributs, der normalisiert werden soll, min_A und max_A sind das Minimum und das Maximum des Wertebereichs von Attribut A, und new_min_A und new_max_A sind das neue Minimum und das neue Maximum des neuen Wertebereichs für A.
2. **Z-Score-Normierung:** $v'(v) = \frac{v-\bar{A}}{\sigma_A}$ mit $v \in dom(A)$ und $\bar{A}$ = Durchschnittswert von Attribut A und σ_A = Standardabweichung von A. Die Z-Score-Normierung ist sinnvoll, wenn entweder das Maximum und das Minimum unbekannt sind oder Ausreißer die Min-Max-Normierung dominieren.
3. **Dezimal-Normierung:** $v'(v) = \frac{v}{10^j}$ mit $v \in dom(A)$ und $j = min\{j \in \mathbb{N} | max(|v'|) < 1\}$ Die Dezimal-Normierung skaliert die Werte in den Bereich $[-1, 1]$.
4. **Skalen-Normierung:** $v'(v) = \frac{v}{max(abs(min_A), abs(max_A))}$ mit $v \in dom(A)$. Die Skalen-Normierung skaliert die Werte v des Attributs A in den Bereich $[-1, 1]$, wobei zur Berechnung das Maximum der Absolutwerte min_A (Minimum von Attribut A) und max_A (Maximum von Attribut A) benutzt wird.

2.5.2 Diskretisierung

Bei der Diskretisierung von Attributen handelt es sich um die Transformation von numerischen auf nominale oder von numerischen auf numerische Attribute, wobei der Wertebereich des Attributs reduziert wird. Die numerischen Werte werden dabei in kleine Mengen von disjunkten Bereichen diskretisiert. Eine Diskretisierung von Attributen ist beispielsweise notwendig, wenn ein Data Mining-Verfahren keine kontinuierlichen Werte verarbeiten kann [HL02][SH00]. Sie ist ebenfalls sinnvoll, wenn die Anzahl an disjunkten Werten des Attributs zu groß ist, da dadurch möglicherweise im Data Mining-Schritt ungenügende Ergebnisse aufgrund der hohen Detaillierung erzielt werden, z. B. die Angabe des Alters in Jahren vs. die Angabe des Alters in die Bereiche $0-25$, $25-40$, $40-65$, und >65.

Im Beispiel in Abb. 2.2 werden die Noten, die in Prozent, also im Bereich zwischen 0% - 100%, angegeben sind, auf die Mengen 1 bis 6 diskretisiert.

Abbildung 2.2: Beispiel: Noten

Doch diese Transformation ist nicht verlustfrei, da potentiell wichtige Details vernachlässigt werden, die jedoch möglicherweise interessant für den Data Mining Prozess gewesen wären. In Abb. 2.2 gehen beispielsweise die genauen Prozente, die einer Note entsprechen, verloren. Womit keine Aussagen mehr getroffen werden können, wie viel Prozent einer Prüfung nur knapp einer besseren respektive schlechteren Note entgangen sind.

Für die Diskretisierung gibt es zwei Ansätze - die überwachte (supervised) und die unüberwachte (unsupervised) Diskretisierung [DKS95]. Bei der überwachten Diskretisierung werden die Daten in den Bildungsprozess der disjunkten Intervalle mit integriert, bei der unüberwachten erfolgt die Intervallbildung ohne jegliche Kenntnis der Daten. Im folgenden soll die Diskretisierung anhand der Binning-Methode näher erläutert werden:

Binning: Unter Binning ist die Zusammenfassung von ähnlichen Werten in Partitionen oder in Bins (Behälter) zu verstehen, wodurch eine Reduzierung von unterschiedlichen Werten erzielt wird. Für das Binning gibt es verschiedene überwachte und unüberwachte Methoden einschließlich den Folgenden:

1. Das Binning erfolgt auf **Bins gleicher Breite** (Equiwidth-Binning). Es handelt sich hierbei um eine recht einfache Technik, die auf numerische Attribute angewendet wird. Dazu wird das Minimum und das Maximum des Attributes bestimmt und dann in `N` Bins mit einer Breite $d = \frac{(Maximum - Minimum)}{N}$ unterteilt. Die Anzahl der Bins wird dabei entweder automatisch bestimmt (überwachte Diskretisierungsmethode) oder vom Benutzer festgelegt (unüberwachte Diskretisierungsmethode).
2. Das Binning erfolgt auf **Bins gleicher Höhe** (Equidepth binning). Diese Technik wird auf numerische Attribute angewendet und versucht die Menge der Objekte in den Bins so auszubalancieren, so dass in jedem Bin ungefähr die gleiche Anzahl ist. Die Breiten der Bins sind entsprechend unterschiedlich groß. Das Equidepth-Binning ist eine überwachte Diskretisierungsmethode.
3. Das Binning erfolgt nach den `Top-N` **häufigsten Werten** (Top-N-Most-Frequent-Items-Binning). Diese Technik wird auf nominale Attribute angewendet und die Bezeichner für die Bins ergeben sich aus der Berechnung der Häufigkeiten der nominalen Werte des Attributes. Dabei werden die `N` häufigsten nominalen Werte als Bezeichner für die Bins benutzt, alle anderen Werte fallen in ein zusätzliches Bin mit der Bezeichnung „andere“. Das Top-N-Frequent-Items-Binning ist eine überwachte Diskretisierungsmethode.

Neben dem Binning können ebenfalls die Histogramm-Analyse, das Clustering (eigentlich eine Data Mining-Methode, die Daten anhand von Kriterien gruppiert (siehe auch 2.6.2)), die Segmentbildung durch natürliche Partitionierung und die Entropie-basierte Diskretisierung zur Diskretisierung verwendet werden [PT98][HL02][DKS95].

2.5.3 Aggregierung

Unter einer Aggregierung ist eine Art Zusammenfassung von Sachverhalten zu verstehen. Diese Zusammenfassungen können mit den Aggregierungsfunktionen - Summe, Durchschnitt, Maximum, u. s. w. erzielt werden. Z. B. Die monatlichen Einnahmen eines Unternehmens werden zusammengefasst zu den jährlichen Einnahmen.

2.5.4 Generalisierung

Bei einer Generalisierung werden einzelne Sachverhalte zusammengefasst, verallgemeinert oder vereinfacht. Eine Generalisierung erfolgt häufig unter Verwendung von Konzept-Hierarchien[4], z. B. können verschiedene Straßentypen (Feldweg, Bundesstraße, Autobahn) zu Straße abstrahiert werden.

2.5.5 Attribut-Konstruktion

Neue Attribute werden aus bestehenden Attributen erzeugt, um beim anschließenden Data Mining-Prozess die Ausdrucksfähigkeit und das Verständnis der Daten zu erhöhen. Der Nutzen konstruierter Attribute ist häufig höher als der Nutzen der bestehenden Attribute, aus denen die neuen Attribute konstruiert wurden. Beispielsweise möchte man das Attribut *Fläche* basierend auf den Attributen *Länge* und *Breite* einführen.

2.5.6 Attribut-Reduktion

Das Data Mining auf riesigen Datenmengen, bezogen auf eine Attributmenge A, kann sehr zeitintensiv sein. Dadurch wird eine Analyse häufig unpraktikabel und in einigen Fällen sogar undurchführbar. Die Datenreduktion durch Attribut-Reduktion kann die Daten von irrelevanten oder nur schwach-relevanten Attributen befreien und sich hauptsächlich auf die wirklich relevanten Attribute beziehen. Dazu muss eine minimale Attributmenge $A' \subset A$ gefunden werden, so dass sich das Ergebnis ähnlich zu dem der kompletten Menge verhält. Durch die Attribut-Reduktion erscheinen weniger Attribute in den gefundenen Mustern und sind dadurch einfacher verständlich. Dabei ist es notwendig eine Unterscheidung zwischen relevanten und nicht-relevanten Attributen zu treffen. Methoden, die zur Attribut-Reduktion verwendet werden können, sind in [CP97] und [Fod02] beschrieben.

2.6 Data Mining

Das Data Mining ist der entscheidende Schritt im KDD-Prozess und beschreibt den eigentlichen Vorgang der Knowledge Discovery. Die Data Mining-Verfahren, die dafür eingesetzt werden, sollen dabei so autonom wie möglich interessante Gesetzmäßigkeiten (Muster) in den Daten identifizieren und extrahieren. Damit das Data Mining erfolgreich verläuft und das jeweilige Data Mining-Verfahren möglichst gute und sinnvolle Ergebnisse in Bezug auf die angestrebten Ziele liefert, wurden die Daten in den vorangegangenen Schritten 1-4 des KDD-Prozesses entsprechend vorbereitet.

[4] Eine Konzept-Hierarchie ist eine Zuordnung von Begriffen auf niedriger Stufe zu Begriffen auf höherer Stufe.

Das Data Mining wird in [FPSS96] folgendermaßen beschrieben: *„Data Mining is a step in the KDD process consisting of applying computational techniques that, under acceptable computational efficiency limitation, produce a particular enumeration of patterns over the data.“*, d. h. Data Mining ist ein Schritt im KDD-Prozess, bei dem unter Anwendung von rechenbetonten Techniken und akzeptabler rechenbetonter Begrenzung der Leitungsfähigkeit eine bestimmte Aufzählung von Mustern produziert wird, die in den Daten enthalten sind.

Das Data Mining ist als eine Zusammensetzung aus mehreren anderen Fachbereichen zu verstehen. Es ist somit ein interdisziplinärer Arbeitsbereich, der die Vorzüge und Errungenschaften der anderen Fachbereiche in einem neuen Bereich zu vereinen versucht. Zu diesen Bereichen gehören nach [Säu00] und [HK00] Datenbanksysteme (Verwaltung großer Datenmengen), Statistik (Mustererkennung und Musterbeschreibung), Maschinelles Lernen (Algorithmen), Künstliche Intelligenz (Algorithmen), Mustererkennung (Algorithmen), Hochleistungsrechentechnik (Verarbeitung großer Datenmengen) und Datenvisualisierung. Übersichtlich dargestellt sind die anderen Fachbereiche in Abb. 2.3.

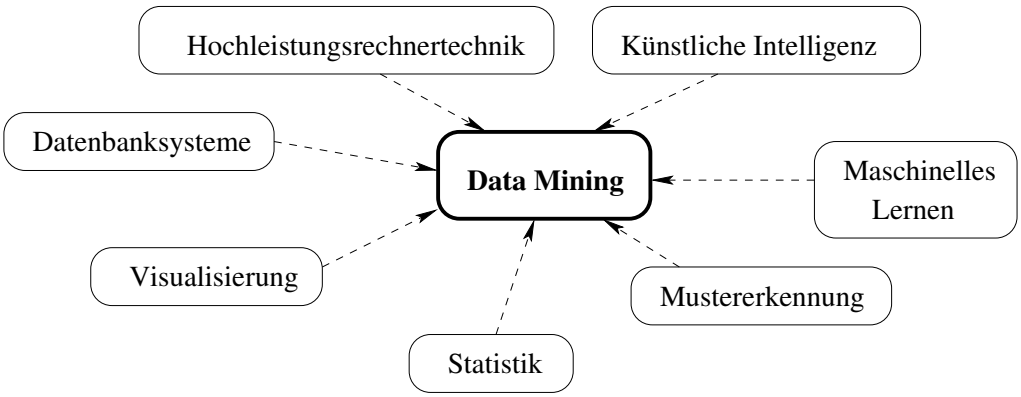


Abbildung 2.3: Data Mining-Bereiche

2.6.1 Übersicht zum Data Mining

Das Data Mining umfasst eine Menge von Methoden, die zur Knowledge Discovery eingesetzt werden können. Diese Data Mining-Methoden lassen sich in deskriptive (unsupervised) und prädiktive (supervised) Methoden unterteilen. Die prädiktiven Methoden funktionieren, allgemein betrachtet, nach dem Prinzip vom Lernen an Beispielen, und im Gegensatz dazu kommen die deskriptiven Methoden ohne Beispiele aus.

Die deskriptiven Methoden charakterisieren die allgemeinen Eigenschaften der Daten. Diese werden vorwiegend eingesetzt, um Informationen aus den bestehenden Daten zu extrahieren. Beispielsweise lassen sich mit deskriptive Methoden Abweichungen, Regelmäßigkeiten oder Ballungen in Daten finden. Die prädiktiven Methoden führen hingegen Schlussfolgerungen auf Daten anhand der Eigenschaften durch und konstruieren damit ein sogenanntes Vorhersage-Modell. Unter Anwendung des Vorhersage-Modells soll dann eine Aussage über neue Daten anhand von gegebenen Eigenschaften gemacht werden können.

Gemeinsam haben alle Verfahren, dass sie Muster (engl. Pattern) und im weiteren Sinne Wissen in großen Datenmengen finden sollen. Dabei stehen besonders Muster im Vordergrund, die zum einen für den Benutzer interessant sind und zum anderen interessanter als triviale Muster sind. Folgende Abgrenzung zwischen einem Muster und Wissen gibt [FPSS96]: *„A pattern that is interesting and certain enough is called knowledge.“*, d. h. ein Muster wird Wissen genannt, wenn es interessant und sicher genug ist.

Die Data Mining-Methoden lassen sich, wie bereits beschrieben, in zwei Gruppen aufteilen, die im Folgenden aufgelistet und in den nächsten Abschnitten beschrieben werden:

- Deskriptiv (Unsupervised)
  - Clustering
  - Assoziationsanalyse
- Prädiktiv (Supervised)
  - Klassifikation
  - Regression

In weiteren Abschnitt werden einige der bekanntesten Data Mining-Methoden vorgestellt und an Beispielen näher erläutert.

2.6.2 Clustering

In Abb. 2.4 sind beispielhaft einige Datenmengen illustriert. Bei näherer Betrachtung dieser Datenmengen ist auffällig, dass es dort Bereiche gibt, in denen sich Daten konzentrieren bzw. gruppieren. Gruppierungen haben die Eigenschaft, dass die darin enthaltenen Daten in irgendeiner weise ähnlich zueinander sind. Das Data Mining unter Verwendung des Clustering versucht solche Gruppierungen zu identifizieren, z. B. die Einteilung von Kunden nach Interessensgebieten, so dass aus den Gruppierungen Schlussfolgerungen gezogen werden können.

Abbildung 2.4: Beispiel-Datenmengen[ME96]

Durch das Zusammenfassen der Daten zu Gruppen, auch Cluster oder Klassen genannt, ergeben sich somit charakteristische Eigenschaften, die die jeweilige Gruppe besonders auszeichnet. Weiterhin können Aussagen über die erzeugten Gruppen und deren Repräsentanten gemacht werden, die im Vorhinein nicht möglich gewesen sind. Das Clustering wird also zur Gruppierung von Daten eingesetzt. Das Ergebnis einer solchen Gruppierung bzw. Clustering ist in Abb. 2.5 dargestellt, wobei zwei unterschiedliche Gruppierungsmethoden, die im folgenden noch erläutert werden, verfolgt wurden – partitioning-based Methode (Abb. 2.5 a) und density-based Methode (Abb. 2.5 b).

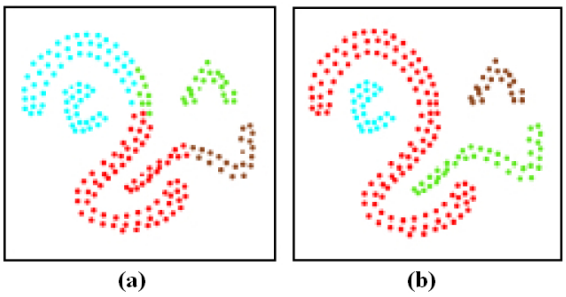


Abbildung 2.5: Clustering Ergebnis[ME96]

Das Clustering stellt ein deskriptives Data Mining-Verfahren dar. Der Prozess des Clustering unterteilt eine Menge von Objekten derart in Klassen oder Clustern, so dass ähnliche Objekte zusammengefasst und unähnliche separiert werden. Das Clustering verwendet dazu eine Ähnlichkeits- bzw. Abstandsfunktion, die abhängig von den betrachteten Attributen und von der betrachteten Clustering-Methode ist. Als Ähnlichkeits- bzw. Abstandsfunktion für räumliche Daten im $\mathbb{R}^n$ können z. B. die folgenden Funktionen verwendet werden:

1. Der **Euklidischer-Abstand** zweier Objekte ist definiert als: $d(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{\sum_i^n (x_i - y_i)^2}$, wobei die Objekte durch zwei n-dimensionale Vektoren $\overrightarrow{x}$ und $\overrightarrow{y}$ repräsentiert werden. Es seien $\overrightarrow{x} = (1,2,1,0,5)$ und $\overrightarrow{y} = (4,3,0,0,3)$ zwei 5-dimensionale Vektoren, dann ergibt der Euklidische-Abstand $d(\overrightarrow{x}, \overrightarrow{y}) \approx 3,6$.

2. Der **Cosinus-Abstand** zweier Objekte ist definiert als: $d(\overrightarrow{x}, \overrightarrow{y}) = 1 - \cos(\overrightarrow{x}, \overrightarrow{y})$, wobei für den Cosinus gilt: $\cos(\overrightarrow{x}, \overrightarrow{y}) = \frac{<\overrightarrow{x}, \overrightarrow{y}>}{||\overrightarrow{x}|| \cdot ||\overrightarrow{y}||}$, ($< \cdot, \cdot >$ entspricht dem Skalarprodukt zweier Objekte und $|| \cdot ||$ entspricht der Länge bzw. dem Betrag eines Objekts) und die Objekte durch zwei n-dimensionale Vektoren $\overrightarrow{x}$ und $\overrightarrow{y}$ repräsentiert werden. Durch den Cosinus-Abstand wird der Korrelationskoeffizient zwischen zwei Vektoren beschrieben. Ein Koeffizient nahe 0 beschreibt eine signifikante Korrelation und nahe 1 keine Korrelation. Es seien $\overrightarrow{x} = (1, 2, 1, 0, 5)$ und $\overrightarrow{y} = (4, 3, 0, 0, 3)$ zwei 5-dimensionale Vektoren, dann ergibt der Cosinus-Abstand $d(\overrightarrow{x}, \overrightarrow{y}) \approx 0, 2$.
3. Der **Schneller-Cosinus-Abstand** zweier Objekte ist definiert als: $d(\overrightarrow{x}, \overrightarrow{y}) = 1 - < \overrightarrow{x}, \overrightarrow{y} >$ und somit ähnlich zum Cosinus-Abstand. Der Unterschied liegt in der Normalisierung der Vektoren $\overrightarrow{x}$ und $\overrightarrow{y}$, da angenommen wird, dass $|| \overrightarrow{x} || = || \overrightarrow{y} || = 1$.

Eine formale Definition für das Clustering gibt [Joa]:

Definition (Clustering): Sei $X = \{o_1, ..., o_n\}$ ein Menge von Objekten. Sei weiterhin $q : \mathcal{P}(\mathcal{P}(X)) \rightarrow \mathbb{R}$ eine Qualitätsfunktion, die die Qualität der erstellten Cluster misst. Für die Definition der Qualitätsfunktion q wird häufig eine Abstandsfunktion (siehe Beispiel weiter hinten) verwendet. Gegeben X und q besteht die Aufgabe des Clustering darin, eine Menge von Clustern $C = \{C_1, ..., C_k\} \in \mathcal{P}(\mathcal{P}(X))$, wobei $(C_i \in \mathcal{P}(X)) \subset X$ für alle $i = \{1, ..., k\}$, zu finden, so dass $q(C)$ maximiert wird, d. h. die Ähnlichkeit der Objekte o_i innerhalb eines Clusters soll maximal sein und die Ähnlichkeit der Objekte o_j in unterschiedlichen Clustern sollen minimal sein. Es gilt dabei $X = \bigcup_{i=\{1,...k\}} C_i$. Weiterhin soll optional gelten, dass die Cluster disjunkt sind, d. h. $C_i \cap C_j = \emptyset$ für alle $i \neq j \in \{1, ..., k\}$.

Die wichtigsten Clustering-Verfahren lassen sich in die folgenden Kategorien (Abb. 2.6) einteilen. Die Wahl des Verfahrens ist dabei häufig abhängig von den zur Verfügung stehenden Daten, da die einzelnen Verfahren auf gleichen Daten unterschiedlich gute Cluster bilden können. Im Folgenden werden die einzelnen Kategorien kurz beschrieben. Des Weiteren werden die wichtigsten und am häufigsten verwendeten Verfahren bzgl. ihrer Kategorie anhand von Beispielen näher erläutert. Für ein tieferes Verständnis aller Kategorien und den damit verbundenen Verfahren sei auf die angegebene Literatur [JMF99][HK00][Sch05] verwiesen.

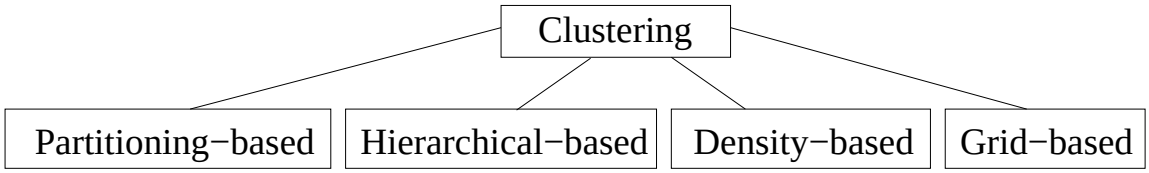


Abbildung 2.6: Kategorien des Clustering

a.) Partitioning-based Methoden

Gegeben seien n Objekte. Weiter sei $k \in \mathbb{N}$ die Anzahl der zu erzeugenden Partitionen. Ebenfalls gegeben sei eine Qualitätsfunktion (Clustering-Kriterium), die die Qualität der Partitionierung misst und auf einer Abstandsfunktion basiert, welche die Ähnlichkeit zwischen jeweils zwei Objekten angibt. Partitionierende Verfahren organisieren diese Objekte in die k Partitionen mit $k \leq n$, wobei jede Partition einem Cluster entspricht. Dabei sind stets die folgenden Bedingungen erfüllt: (1) Jeder Cluster muss mindestens ein Objekt enthalten und (2) jedes Objekt muss genau einem Cluster zugeordnet sein. Diese Methode startet häufig mit einer initialen Partitionierung der n Objekte und versucht anschließend iterativ, die Partitionierung derart zu optimieren, dass der Wert der Qualitätsfunktion maximiert wird. Ein Nachteil dieser Methode ist, dass nur konvexe Cluster[5] gefunden werden.

[5]Eine geometrische Menge ist konvex, wenn die Verbindungsstrecke zweier beliebiger Punkte in der Menge liegt.

Unter den partitioning-based Methoden befinden sich, z. B. das *k-Means*-Clustering, das *k-Medoids*-Clustering und deren Variationen. Diese Verfahren gehören zu den einfachsten und weit verbreitetsten. Sie gelten als die klassischen Vertreter des Clustering. Eine frühe Implementierung der *k-Means* und *k-Medoids*-Algorithmen sind der Algorithmus von [Mac67] und PAM (Partitioning Around Medoids) von [KR87]. Eine Weiterentwicklung, die besser mit größeren Datenmengen umgehen konnte, ist das sample-basierte Verfahren CLARA (Clustering LARge Applications) [KR90]. Die Qualität und Skalierbarkeit von CLARA wurde anschließend noch mit dem Verfahren CLARANS (Clustering Large Applications based on RANdomized Search) [NH94][NH02] gesteigert.

Beispiel einer Partitioning-based Methode:

Das partitioning-based Clustering soll anhand des bekannten und weit verbreiteten *k*-Means-Verfahrens gezeigt werden. Es sei folgende Datenmenge X gegeben: X = $\{o_1, \ldots, o_n\}$ = {(1,1); (1,2); (2,2); (3,1); (1.5,1.5); (3.5,0.5); (4,1); (3.25, 2); (3,4)}. In Abb. 2.7 (a) ist diese Menge graphisch in einem Diagramm aufgetragen. Weiter sei die Anzahl der Cluster durch K und der Mittelpunkt (*Mean*) des Clusters k durch $M_k := \frac{\sum_{i=1}^{n} o_i}{n}$ mit $o_i \in \mathbb{R}^2$ gegeben. Der Mittelpunkt ergibt sich aus dem Durchschnitt der n Objekte im Cluster, wobei die Objekte o_i komponentenweise betrachtet werden. Für die Qualitätsfunktion q wird das Squared-Error-Kriterium [JMF99] verwendet:

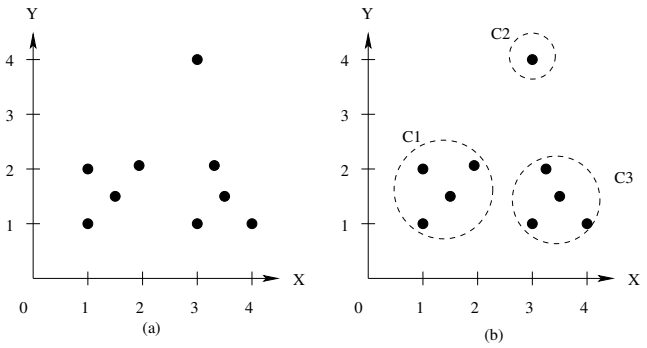


Abbildung 2.7: Beispiel: k-Means

$$q(\{C_1, \ldots C_k)\}) := se(K, X) = \sum_{j=1}^{K} \sum_{i=1}^{o_j} (|o_i^{(j)} - M_j|^2)$$

Dabei handelt es sich bei $o_i^{(j)}$ um das i-te Objekt im j-ten Cluster und bei M_j um den Mittelpunkt im j-ten Cluster. (*Anmerkung:* Die innere Summe des Squared-Error-Kriteriums berechnet die quadratische Differenz für ein einzelnes Cluster j und die äußere Summe fasst die Ergebnisse der einzelnen Cluster zu einem Wert zusammen). In der Qualitätsfunktion q ist implizit das Quadrat der Euklidische-Norm $d(o_i, M_j) = \sqrt{(o_i - M_j)^2}$ mit $o_i, M_j \in \mathbb{R}^2$ als Abstandsfunktion enthalten.

Das *k*-Means-Verfahren soll die Datenmenge X in drei Cluster ($K = 3$) partitionieren. Die initialen Mittelwerte der Cluster C_1, C_2 und C_3 seien zufällig mit $M_1 = (1, 1)$, $M_2 = (2, 2)$ und $M_3 = (4, 1)$ gewählt. Das Clustering ist in Abb. 2.8 anschaulich illustriert, wobei die Cluster C_i und die zugeordneten Daten und die Mittelpunkte M_i (grüne Punkte in Abb. 2.8) angegeben sind.

> **Funktionsweise:** Es werden alle Daten nacheinander durchgegangen und jeweils die Abstände zu den Mittelpunkten der Cluster berechnet. Das aktuell betrachtete Objekt wird dann dem Cluster zugeordnet, zu dessen Mittelpunkt das Objekt den minimalsten Abstand hat. Gibt es keine Änderungen der Cluster in der i-ten Iteration im Vergleich zur $(i-1)$-ten Iteration, so endet der Algorithmus. Der Algorithmus endet ebenfalls, falls eine festgelegte Anzahl an Iterationen durchgeführt wurde.

Der Algorithmus endet nach vier Iterationen, da keine weiteren Verschiebungen unterhalb der Cluster zwischen der 3. und 4. Iteration stattfinden (aus diesem Grund wurde die 4. Iteration nicht mehr veranschaulicht). Das Ergebnis vom k-Means-Verfahren ist im letzten Diagramm in Abb. 2.8 zu sehen. Außerdem spielen die Wahl der Initial-Mittelwerte und die Anzahl der zu erzeugenden Cluster eine entscheidende Rolle in puncto Anzahl der Iterationen und Qualität der Cluster [Dub87].

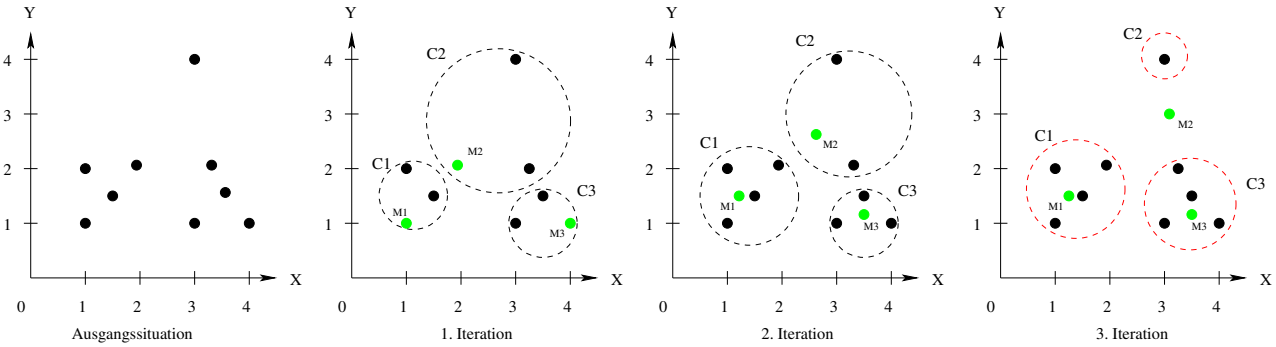


Abbildung 2.8: Partitioning-based Clustering: k-Means

b.) Hierarchical-based Methoden

Hierarchische Verfahren gruppieren Objekte in eine Hierarchie von Clustern, in der die Ebenen die Ähnlichkeit der Objekte wiederspiegelt. Dabei unterscheiden sich die Verfahren in *agglomerative* und *divisive*, je nachdem ob die hierarchische (De-)komposition bottom-up oder top-down ist.

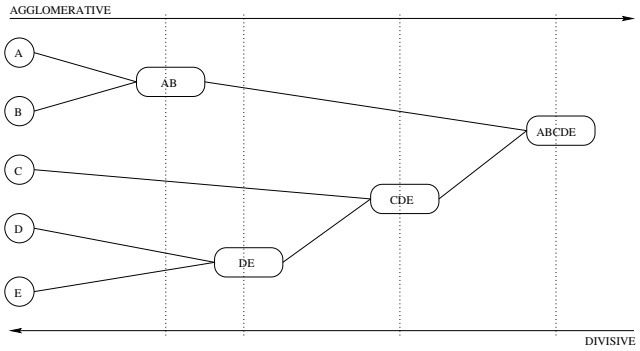


Abbildung 2.9: Clustering der Daten {A,B,C,D,E}

Der agglomerative Ansatz startet mit jedem Objekt in einem eigenen Cluster, und fasst diese anhand eines Clustering-Kriteriums sukzessive zusammen bis alle Objekte in einem einzigen Cluster zusammengefasst sind oder ein Haltekriterium erreicht ist. Der divisive Ansatz startet mit allen Objekten in einem Cluster und teilt den Cluster anhand eines Clustering-Kriteriums in kleinere Cluster auf bis alle Objekte in einem eigenem Cluster sind oder ein Haltekriterium erreicht ist.

Das Kriterium, wann zwei oder mehrere Cluster zu einem neuen Cluster zusammengefügt werden (agglomerativ) bzw. wann ein Cluster in zwei oder mehrere Cluster aufgespalten wird (divisiv), ist i. d. R. durch eine Abstandsfunktion (Ähnlichkeitsfunktion) gegeben, die den Abstand zwischen den Clustern angibt. Daraus ergibt sich implizit das Clustering-Kriterium (Qualitätsfunktion q), das die Qualität der Cluster misst. Die Qualitätsfunktion q entscheidet demnach welche Aufteilung eines Clusters in mehrere kleinere Cluster bzw. welche Zusammenfassung von Clustern zu einem größeren Cluster den größten Nutzen bringt, in dem die Qualität der neuen Cluster gemessen wird.

In der Literatur gibt es dafür zwei gängige Ansätze - 1. *Single-Link Methode* (d. h. kleinster Abstand zwischen einem Element in einem Cluster und einem Element in einem anderen Cluster) und 2. *Complete-Link Methode* (d. h. größter Abstand zwischen einem Element in einem Cluster und einem Element in einem anderen Cluster). Das Haltekriterium ist sowohl bei den agglomerativen als auch bei den divisiven Methoden häufig eine maximale Abstandsangabe zwischen zwei Clustern oder die Anzahl der Cluster, die erzeugt werden sollen.

Die Hierarchie der gebildeten Cluster wird durch ein sogenanntes *Dendrogramm* repräsentiert. Ein Beispiel für ein Dendrogramm zeigt Abb. 2.9, in dem die Daten {A,B,C,D,E} geclustert werden. Nach der Generierung des Dendrogramms durch das Clustering-Verfahren kann dieses, je nach dem wie viele Cluster benötigt werden, auf verschiedenen Ebenen der Hierarchie aufgebrochen werden. Die Qualität von hierarchischen Clustern leidet jedoch darunter, dass einmal vorgenommene Splits/Merges nicht rückgängig gemacht werden können. Algorithmen, die zu den hier-

archischen Algorithmen gehören, sind DIANA (DIvisive ANAlysis) und AGNES (AGglomerative NESting)[KR90]. Weitere Algorithmen z. B. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)[TZ96] und CHAMELEON [KHN99] sind ebenfalls hierarchische Clustering-Methoden, die jedoch versuchen unter Benutzung anderer Clustering-Methoden (partitioning-based Methoden) die Qualität der Cluster zu erhöhen.

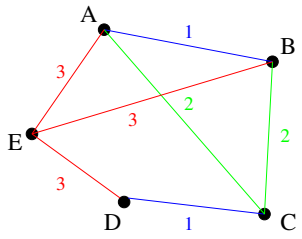
Beispiel für eine Hierarchical-based Methode:

Das hierarchical-based Clustering soll anhand einer agglomerativen Methode demonstriert werden. Dabei werden Cluster C_i erzeugt, indem jeweils kleinere Cluster zu größeren Clustern zusammengefasst werden. Als Kriterium zum Verschmelzen von Clustern sei die Single-Link Methode gewählt und es sei weiter kein zusätzliches Haltekriterium angegeben, das das Clustering frühzeitig beenden kann.

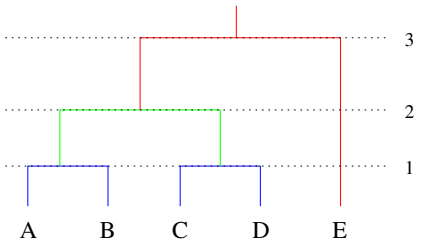
Gegeben sei die in Abb. 2.10 (a) dargestellte Datenmenge, bestehend aus den Objekten A, B, C, D, E. Die Objekte sind in einer Ähnlichkeitsmatrix [HK00] angeordnet, die die Ähnlichkeit also den Abstand (z. B. euklidischer Abstand bei numerischen Werten oder lexikalischer Abstand bei nominalen Werte) der Objekte zueinander definiert.

Objekt	A	B	C	D	E
A	0	1	2	2	3
B		0	2	4	3
C			0	1	5
D				0	3
E					0

(a)



(b)



(c)

Abbildung 2.10: Beispiel für hierarchisches Clustering mit Single-Link

Die Clustering-Methode fasst die Cluster sukzessive (graphisch verdeutlicht in Abb. 2.10) (b) anhand der Single-Link Methode zusammen, bis sich alle Objekte in einem einzigen Cluster befinden. Die Ebenen des Dendrogramms beschreiben die Ähnlichkeit zwischen den Clustern, d. h. je niedriger die Ebene auf der zwei Cluster zusammengefasst werden, um so ähnlicher die Objekte in diesen Clustern. Die aus dem Algorithmus resultierende Cluster-Hierarchie in Form eines Dendrogramms ist in Abb. 2.10 (c) dargestellt.

c.) Density-based Methoden

Density-based Verfahren produzieren Cluster beliebiger Form, z. B. kreisförmige, gekrümmte, längliche geometrische Formen. Die Idee dahinter ist, Cluster solange wachsen zu lassen, bis die Dichte (Anzahl der Objekte in einem bestimmten Bereich) einen Grenzwert erreicht hat. Die Cluster stellen Gebiete dar, in denen die Objekte dicht beieinander liegen und durch Gebiete getrennt sind, in denen die Objekte weniger dicht liegen. Dadurch ergeben sich die wichtigsten Merkmale von density-based Clustern: (1) Jeder Cluster weißt eine definierte Dichte von Punkten auf und (2) die Dichte innerhalb eines Clusters ist höher als die Dichte außerhalb. Typische density-based Algorithmen sind DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [ME96] und OPTICS (Ordering Points To Identify the Clustering Structure) [AMJ99], der eine Verbesserung zu DBSCAN darstellt.

Beispiel einer Density-based Methode:

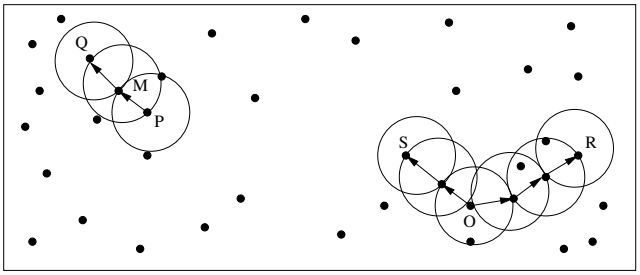


Abbildung 2.11: Beispielmenge [ME96]

Das Clustering mit einer density-based Methode soll anhand des DBSCAN-Algorithmus erläutert werden. Der Kernpunkt des Algorithmus besagt, dass es für jedes Objekt eines Clusters innerhalb eines definierten Radius eine Mindestanzahl von Objekten gibt. Es bedeutet, dass die Dichte im Bereich, der vom Radius um das Objekt definiert wird, einen zuvor definierten Schwellenwert übersteigt. DBSCAN verwendet intern zur Berechnung eine Abstandsfunktion, z. B. Euklidische Norm, die jedoch beliebig variiert werden kann. Weiter sind für die Erklärung der density-based Clustering-Methode weitere essentielle Definitionen notwendig:

- Die Umgebung eines Objektes p mit dem Radius ϵ wird als **ϵ-Umgebung** von p (kurz: $U_\epsilon(p)$) bezeichnet.
- Ein Objekt p wird als **Kern-Objekt** bezeichnet, wenn sich in der ϵ-Umgebung um das Objekt eine Mindestanzahl von Objekten (kurz: MinObj) befinden.
- Ein Objekt p ist **directly-density-reachable** von einem Objekt q, wenn (1.) $p \in U_\epsilon(q)$ und (2.) $|U_\epsilon(q)| \geq MinObj$. D. h. ein Objekt p muss sich in einer ϵ-Umgebung von einem Objekt befinden, die den vorgegebenen Schwellenwert übersteigt. (Es handelt sich dabei um keine symmetrische Beziehung, d. h. q ist directly-density-reachable von p impliziert nicht zwangsläufig p ist directly-density-reachable q. In Abb.2.11 beispielsweise die Objekte P und M.
- Ein Objekt p ist **density-reachable** von einem Objekt q, wenn eine Kette von Objekten $p = o_1, \ldots, o_n = q$ existiert, so dass p_{i+1} **directly-density-reachable** vom p_i ist. D. h. die Objekte p, q mit $p = o_1$ und $q \neq o_2$ sind nicht directly-density-reachable, jedoch existiert eine Kette von Punkten o_i, über die die Objekte p und q miteinander verbunden sind. In Abb. 2.11 beispielsweise die Objekte P und Q.
- Ein Objekt p ist **density-connected** mit einem Objekt q, wenn es ein Objekt o gibt, das mit p und q **density-reachable** ist. D. h. die Objekte p und q besitzen in diesem Fall in ihrer ϵ-Umgebung nicht die geforderte Mindestanzahl an Objekten, sind jedoch über ein anderes Objekt o derart verbunden, dass sowohl p als auch q density-reachable verbunden sind. In Abb. 2.11 beispielsweise die Objekte S und R mit dem Objekt O.

Zusammenfassend lässt sich ein density-based Cluster als eine Menge von density-connected Objekten beschreiben. Jedes Objekt, das nicht in einem Cluster ist, wird als Ausreißer bezeichnet. Formal lässt sich der density-based Cluster wie folgt beschreiben: Ein Cluster C ist eine nicht-leere Menge von Objekten für die gilt: (1) $\forall p, q$: Wenn $p \in C$ und q ist density-reachable von P, dann ist $q \in C$ und (2) $\forall p, q \in C$: p ist density-connected mit q.

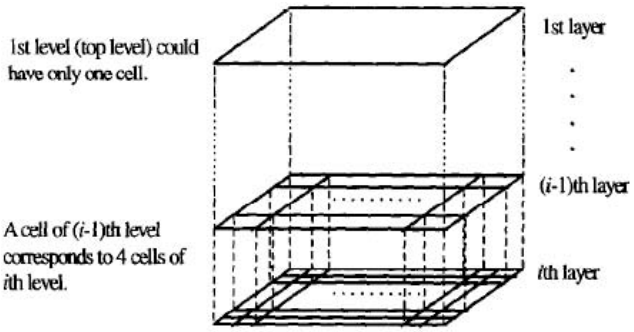
Für den Ablauf des Algorithmus werden als Parameter (1.) die Definition der ϵ-Umgebung und (2.) die Angabe der Dichte, also der MinPts in einer ϵ-Umgebung angegeben. Die Qualitätsfunktion q zur Bewertung der Cluster basiert auf diesen angegeben Parametern, die entscheiden wie die Cluster konstruiert werden. Zunächst werden Objekte in den Daten gesucht, deren ϵ-Umgebung die vorgegebene Dichte aufweisen. Diese Objekte werden als Cluster definiert. Anschließend werden iterativ weitere Objekte zu diesen Clustern hinzugefügt, falls diese directly-density-reachable sind. Die Bewertung der entstehenden Cluster durch die Qualitätsfunktion q erfolgt also stetig mit der Erweiterung der Cluster um neue Objekte. Der Algorithmus endet, falls keine Objekte mehr irgendeinem Cluster hinzugefügt werden können.

d.) Grid-based Methoden

Grid-based Verfahren quantisieren den Datenraum in eine endliche Menge von Zellen, die einer Rasterform (Gitter) entspricht. Alle Operationen werden auf dem Raster durchgeführt, was einen Vorteil bei der Verarbeitung (durch Parallelisierung) bringt, da es nicht von der Anzahl der zu verarbeiteten Objekt abhängig ist, sondern von der Anzahl der Zellen. Ein typischer grid-based Algorithmus ist STING (STatistical INformation Grid) [WYM97].

Beispiel einer Grid-based Methode:

Die Idee des Clustering mit der grid-based Methode soll anhand des STING-Algorithmus erläutert werden. Der STING-Algorithmus unterteilt den Datenraum in eine Menge von disjunkten Zellen, die ein Raster formen. Diese Raster existieren auf verschiedenen Ebenen, die sich lediglich in der Anzahl der Zellen unterscheiden, und formen gesamtheitlich eine hierarchische Struktur (Abb. 2.12). (Auf oberster Ebene bilden wenig Zellen das Raster und auf unterster Ebene bilden viele Zellen das Raster.) Zu jeder Zelle in der Hierarchie werden in einem Preprocessing-Schritt statistische Informationen (z. B. die Dichte der Zelle) gesammelt, die im weiteren Verlauf benutzt werden.


1st level (top level) could have only one cell.
A cell of (i-1)th level corresponds to 4 cells of ith level.
1st layer
(i-1)th layer
ith layer
Abbildung 2.12: Raster Hierarchie[WYM97]

Der STING-Algorithmus verfolgt einen Top-Down-Ansatz und betrachtet und bewertet die Zellen anhand von statistischen Informationen. Die Zellen des Rasters werden sukzessive unterteilt, d. h. in mehrere Zellen aufgeteilt bis die unterste Hierarchie-Stufe erreicht ist. Eine Zelle die also in der i-ten Hierarchie-Stufe interessant ist (gemessen an den statistischen Informationen), wird in der (i+1)-ten Hierarchie-Stufe weiter betrachtet, sonst nicht. Ist die unterste Stufe erreicht werden alle benachbarten Zellen, die beispielsweise eine bestimmte Dichte aufweisen, zu Clustern zusammengefasst, wobei sich Cluster auch auf höheren Stufen der Hierarchie aufgrund der Zusammenfassung ergeben können.

2.6.3 Assoziationsanalyse

In der folgenden Tabelle sind beispielhaft Datensätze eines Supermarktes aufgelistet, die den Inhalt von Warenkörben beschreiben. Bei genauer Analyse der Daten ist auffällig, dass es Artikel gibt, die oftmals zusammen gekauft werden. Dieser Aspekt suggeriert eine Abhängigkeit unter den Artikeln, z. B. Produkt `Brot` wird oftmals mit Produkt `Erdnuss-Butter` gekauft.

Beispiel: Supermarkt

Warenkorb	Bier	Brot	Erdnuss-Butter	Marmelade	Milch
w_1		X	X	X	
w_2		X	X		
w_3		X	X		X
w_4	X	X			
w_5	X				X
w_6		X	X		X

Diese Abhängigkeiten unter den Daten sind von besonderem Interesse, da diese im Allgemeinen nicht offensichtlich sind und somit interessante Erkenntnisse bieten oder Rückschlüsse auf Zusam-

menhänge zulassen. Die Assoziationsanalyse [AMS+96] ist eine Data Mining-Methode mit der diese interessanten und signifikanten Abhängigkeiten (Assoziationen) in den Daten entdeckt werden können. Als typisches Beispiel der Assoziationsanalyse ist die Warenkorbanalyse, die im Beispiel angedeutet ist und die Untersuchung des Kundenverhaltens in Bezug auf die gekauften Artikel beschreibt, zu nennen.

Die Assoziationsanalyse ist eine deskriptive Data Mining-Methode, mit dessen Hilfe Muster mit einfacher Struktur in den Daten gefunden werden können und immer dann Anwendung findet, wenn die Maxime gilt: „*If you don't know what you are looking for, use association analysis.*"

Bei dem Prozess der Assoziationsanalyse werden interessante Assoziationen oder Korrelationen zwischen Daten gefunden, die in *Association Rules* (Assoziationsregeln) ausgedrückt werden. Eine Assoziationsregel ist eine Implikation $A \Rightarrow B$ (d. h. aus A folgt B), z. B. die Assoziation zwischen den Produkten `Brot` und `Erdnuss-Butter` wird als `Brot` ⇒ `Erdnuss-Butter` dargestellt, die besagt, dass wenn der Artikel `Brot` gekauft wird, dann wird auch der Artikel `Erdnuss-Butter` gekauft.

Assoziationsregel: `Brot` ⇒ `Erdnuss-Butter`

Was zeichnet jedoch eine Assoziation gegenüber anderen Assoziationen aus, sind alle gleich interessant oder sind einige interessanter? Aus diesem Grund muss eine Assoziation in irgendeiner Weise bewertet werden, um die Relevanz oder Wichtigkeit einstufen zu können. Diese Bewertung beruht zum einen auf der Häufigkeit (auch *Support* genannt), z. B. in 10% der Fälle wurden `Brot` und `Erdnuss-Butter` zusammen gekauft, und zum anderen auf dem Vertrauen einer Assoziation (auch *Confidence* genannt), d. h. wie häufig trifft die Assoziation zu, z. B. in 70% der Fälle, in denen `Brot` gekauft wurde, wurde ebenfalls `Erdnuss-Butter` gekauft. Die Assoziationsregel `Brot` ⇒ `Erdnuss-Butter` lässt sich mit den Zusatzinformationen über die Relevanz beziehungsweise Wichtigkeit folgendermaßen schreiben:

Assoziationsregel: `Brot` ⇒ `Erdnuss-Butter` [Support=10%, Confidence=70%]

Die Assoziationsanalyse, die eine Data Mining-Methode für die Analyse von Abhängigkeiten in den Daten beschreibt, lässt sich formal wie folgt definieren:

Definition (Assoziationsanalyse): Sei I eine Menge von Objekten (häufig *Items* genannt) und T eine Menge von Transaktionen, wobei gilt: $\forall t \in T : t \subseteq I$. Sei weiterhin $s \in [0\%, 100\%]$ ein Support und $s_{min} \in [0\%, 100\%]$ ein benutzerdefinierter Minimal-Support, und $c \in [0\%, 100\%]$ eine Confidence und $c_{min} \in [0\%, 100\%]$ eine benutzerdefinierte Minimal-Confidence. Bei der Assoziationsanalyse sind alle Assoziationsregeln zu finden, die folgende Bedingungen erfüllen:

1. Form: $A \Rightarrow B$ mit $A \subseteq I$ und $B \subseteq I$, und $A \cap B = \emptyset$
2. Support: $s(A \Rightarrow B) = \frac{|\{t \in T : A \cup B \subseteq t\}|}{|T|} \geq s_{min}$
3. Confidence: $c(A \Rightarrow B) = \frac{s(A \Rightarrow B)}{s(A)} = \frac{|\{t \in T : A \cup B \subseteq t\}|}{|\{t \in T : A \in t\}|} \geq c_{min}$

Die oben definierten Bedingungen drücken aus, wann eine Assoziationsregel gültig und interessant ist. Die erste Bedingung gibt lediglich die Form vor, in der eine Assoziationsregeln dargestellt wird. Die zweite Bedingung gibt an, dass eine gewisse Minimalhäufigkeit für die in der Assoziationsregel vorkommenden Objekte gegeben sein muss, d. h. es sind nur Assoziationsregeln interessant, die eine bestimmte Relevanz, ausgedrückt durch die Häufigkeit des Auftretens, aufweisen können. Schlussendlich gibt die dritte Bedingung an, dass Objekte mit einer bestimmten Häufigkeit zusammen auftreten müssen, d. h. Transaktionen $t \in T$, die die Prämisse A enthalten, müssen mindestens einen Anteil von c_{min} in der Konklusion B beinhalten.

Eine Assoziationsregel, die einen Minimal-Support $s \geq s_{min}$ und eine Minimal-Confidence $c \geq c_{min}$ erfüllt, wird **Strong Association Rule** (starke Assoziationsregel) genannt. Die starken Assoziationsregeln sind häufig diejenigen, die später an den Benutzer weitergegeben werden, da ansonsten die

Gefahr besteht, dass zu viele unwichtige und triviale Assoziation ermittelt werden. Typische Werte für den minimalen Support s_{min} sind 0.01=1% und der minimalen Confidence c_{min} sind 0.5=50% [Joa] (allgemein: kleiner Wert für den Support und großer Wert für die Confidence).

Assoziationsregeln können verschiedene Formen annehmen, wobei überwiegend zwischen eindimensionalen und mehrdimensionalen unterschieden wird. Bisher wurden sie lediglich in einer Dimension, also eindimensional, betrachtet. Beispielsweise kann die Warenkorbanalyse durch das Prädikat `enthält()` beschrieben werden, d. h. eine Assoziationsregeln hat die veränderte Form `enthält(X, 'Brot')` $\Rightarrow$ `enthält(X, 'Erdnuss-Butter')`, wobei `X` für einen Warenkorb steht.

- eindimensionale Assoziationsregeln: Diese Form beinhaltet lediglich ein Prädikat (Attribut) und findet, wie bereits angedeutet in der Warenkorbanalyse Anwendung.

  `enthält(X, 'Brot')` $\Rightarrow$ `enthält(X, 'Erdnuss-Butter')`.

- mehrdimensionale Assoziationsregeln: Diese Form beinhaltet mehrere Prädikate (Attribute), z. B. kann eine Assoziation zwischen dem Attribut `Region`, dem Attribut `Bevölkerung` und dem Attribut `Verkehr(saufkommen)` damit beschrieben werden. Sie findet Anwendung in relationalen Datenbanken, in denen Assoziationen zwischen den einzelnen Attributen einer Tabelle beschrieben werden können. Folgendes Beispiel verdeutlicht den Unterschied zur eindimensionalen Assoziationsregel:

  `Region(X, 'Hannover')` $\wedge$ `Bevölkerung(X, '>500000')` $\Rightarrow$ `Verkehr(X, 'hoch')`

Weitere Formen von Assoziationsregeln sind durch Einführung von Konzept-Hierarchien möglich. In Abb. 2.13 sei eine vereinfachte Konzept-Hierarchie für den Warenkorb abgebildet.

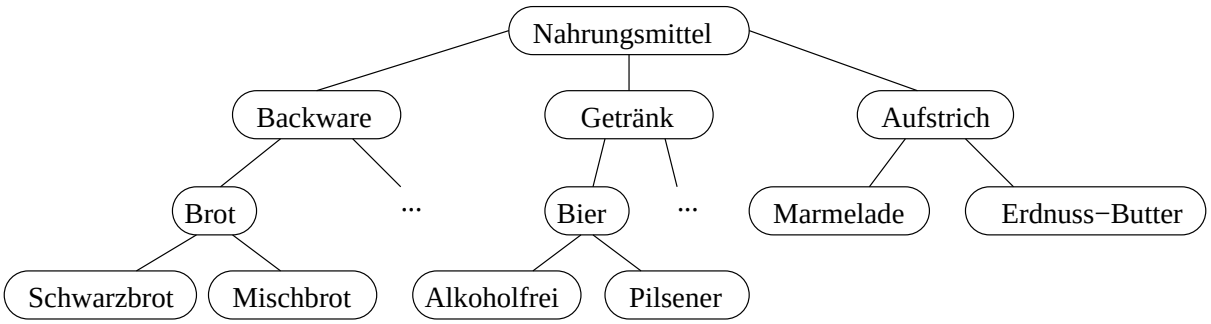


Abbildung 2.13: Konzept-Hierarchie: Warenkorb

Die Konzept-Hierarchien bieten den Vorteil, dass Assoziationsregeln durch deren Verwendung zu starken Assoziationsregeln werden, d. h. den Minimum-Support und die Minimum-Confidence erreichen. Es wird dabei unter den single-level und den multi-level Assoziationsregeln unterschieden:

- single-level Assoziationsregeln: Diese Form der Assoziationsregel beschränkt sich auf eine Abstraktionsebene einer Konzept-Hierarchie, wodurch möglicherweise nur ein Bruchteil an Korrelationen in den Daten gefunden wird oder die Assoziationsregeln zu allgemein sind. Sei beispielsweise ein Minimal-Support $s_{min} = 20\%$ und Minimal-Confidence $c_{min} = 70\%$ gegeben, dann wird die Assoziationsregel, die auf einer niedrigen Ebene der Konzept-Hierarchie fußt, nicht gefunden:

  `kauft(X, 'Schwarzbrot')` $\Rightarrow$ `kauft(X, 'Pilsener')` [s=10%, c=60%].

- multi-level Assoziationsregeln - Die Grundidee dieser Assoziationsregeln besteht darin, dass interessante Zusammenhänge eventuell auch auf anderen Abstraktionsebenen einer Konzept-Hierarchie existieren können [HF95]. Der Nachteil ist, dass diese Assoziationsregeln tendenziell weniger interessant sind, viele ähnliche Zusammenhänge gefunden werden und die zu durchsuchende Menge exponentiell mit der Höhe der Abstraktionsebenen wächst.

Beispielsweise werden Assoziationsregeln mit einem Minimal-Support $s_{min} = 20\%$ und einer Minimal-Confidence $c_{min} = 50\%$ gesucht. Es sei die folgende Assoziationsregel gegeben, die auf einer niedrigen Abstraktionsebene einen Support $s = 10\%$ und Confidence $c = 50\%$ besitzt: `Alter(X, '20..25')` $\Rightarrow$ `kauft(X, 'Alkoholfrei') [s=10%, c=50%]` Wird diese Assoziationsregel jedoch auf einer höheren Abstraktionsebene betrachtet, mit den Abstraktionen von `'20..25' => 'jung'` und `'Alkoholfrei' => 'Bier'`, werden die vorgegebenen Minimalvoraussetzungen erfüllt und die Assoziationsregel gilt als interessant: `Alter(X, 'jung')` $\Rightarrow$ `kauft(X, 'Bier') [s=30%, c=60%]`.

Zur Suche nach Assoziationsregeln wird der Apriori-Algorithmus [AMS+96] eingesetzt, dessen Name auf dem Fakt basiert, dass *prior Knowledge* (voriges Wissen) bei der Suche benutzt wird. Der Apriori-Algorithmus verfolgt dabei einen iterativen Ansatz, bei dem häufig vorkommende Daten als *frequent itemsets* (häufige Mengen) betrachtet werden, und dann die *k-itemsets* (häufige Mengen mit *k* Elementen) benutzt werden, um nach *(k+1)-itemsets* zu suchen. Eine Menge gilt als *frequent itemsets*, wenn diese den definierten Minimal-Support für die Assoziationsregel übersteigt. Der Apriori-Algorithmus macht sich dabei die Eigenschaft „Alle nicht-leeren Untermengen von frequent itemsets sind ebenfalls frequent.“ bzw. „ Wenn eine Menge nicht frequent ist, so sind alle deren Obermengen wiederum nicht frequent. “ zu nutze, und reduziert damit den Suchraum.

Fortsetzung Beispiel: Warenkorb

Unter Anwendung des Apriori-Algorithmus auf die Datenmenge aus diesem Beispiel (s. o.) werden iterativ folgende *frequent itemsets* ermittelt. Beispielsweise ergibt sich für die Menge $K :=$ {`Brot`, `Milch`}, dass die Artikel `Brot` und `Milch` zusammen in zwei von sechs Transaktionen enthalten sind. Bei dieser Menge handelt sich also um ein frequent itemset, da der Support von 33% erfüllt ist. Eine andere Menge $K :=$ {`Marmelade`} ist hingegen nur in einer von sechs Transaktionen enthalten und erreicht lediglich einen Support von 16%. Das hat zur Folge, dass diese Menge in den nachfolgenden Iterationen nicht mehr betrachtet werden muss. Die *frequent itemsets* weisen einen Minimal-Support von 30% auf.

Iteration	Kandidaten [Support%]	Frequent Itemsets
1	{Bier} [33%], {Brot} [66%], {Marmelade} [16%], {Milch} [50%], {Erdnuss-Butter} [66%]	{Bier} , {Brot} , {Milch} {Erdnuss-Butter}
2	{Bier, Brot} [16%], {Bier, Milch} [16%], {Brot, Milch} [33%], {Brot, Erdnuss-Butter} [66%] {Bier, Erdnuss-Butter} [0%], {Erdnuss-Butter, Milch} [33%]	 {Brot, Milch} {Brot, Erdnuss-Butter} {Erdnuss-Butter, Milch}
3	{Brot, Erdnuss-Butter, Milch} [33%]	{Brot, Erdnuss-Butter, Milch}

Aus den *frequent itemsets* mit mind. zwei Elementen lassen sich anschließend die Assoziationsregeln ableiten. Die gefundene Beziehung zwischen `Brot` und `Erdnuss-Butter` beispielsweise lässt sich also wie folgt als Assoziationsregeln ausdrücken.

Regeln(`Brot`,`Erdnuss-Butter`) = {

`{Brot` $\Rightarrow$ `Erdnuss-Butter [s=66%, c=75%]} ;`

`{Erdnuss-Butter` $\Rightarrow$ `Brot [s=66%, c=100%]}`

}

Obwohl die Assoziationsanalyse eine probate Technik zum Data Mining darstellt, weist diese auch Schwächen auf. Assoziationsregeln gelten nach der Definition als interessant, wenn die Regel einen

Minimal-Support und eine Minimal-Confidence aufweist, es sich also um eine starke Assoziationsregel handelt. Die Regel wird dann vom Data-Mining-System an den Benutzer weitergereicht. Jedoch besteht häufig das Problem, dass viele derart gefundene Regeln relativ uninteressant für den Benutzer sind, und demzufolge eine enorme Benutzerinteraktion zur Post-Analyse der Assoziationsregeln nach sich zieht. Dieses Problem der Assoziationsanalyse lässt sich nur schwer automatisieren, da die Interessantheit beziehungsweise Wichtigkeit einer Regeln ein subjektives Empfinden ist, und von Benutzer zu Benutzer abweichen kann.

2.6.4 Klassifikation

Beispiel: Ein Kreditinstitut hat in letzter Zeit vermehrt Fälle von Kreditkartenmißbrauch verzeichnet. Das Kreditinstitut möchte aus diesem Grund ein System einsetzen, das seine Kunden besser schützt, indem Betrugsversuche frühzeitig erkannt werden. Wie kann man jedoch entscheiden, wann ein Betrugsfall vorliegt und wann nicht? Für die Entscheidungsfindung wird ein Vorhersagesystem eingesetzt, das anhand von gegebenen Eigenschaften (Attributen) eine Einstufung oder Einteilung in *Betrug* und *kein Betrug* vornimmt. Für die Einteilung von Daten in verschiedene Kategorien oder Klassen, deren Anzahl endlich ist, kommt die Klassifikation zur Anwendung.

Die Klassifikation ist eine prädiktive Data Mining-Methode, die eine bereits vorhandene Klassifizierung von einigen Objekten in den zu analysierenden Daten voraussetzt. Damit soll versucht werden, ein Modell zu finden, das eine allgemeingültige Klassifizierungsvorschrift darstellt. Das Modell enthält die Attribute, sogenannte Prädikatoren, und die Beziehungen zwischen den Attributen, die zur Klassifikation essentiell sind. Es wird zwischen binärer und mehrwertiger Klassifikation unterschieden. Im Beispiel des Kreditkartenmißbrauchs handelt es sich um eine binäre Klassifikation mit den Klassen `Ja` für *Betrug* und `Nein` für *kein Betrug*. Die Klassifikation unterteilt sich in zwei Abschnitte, die in Abb. 2.14 dargestellt sind.

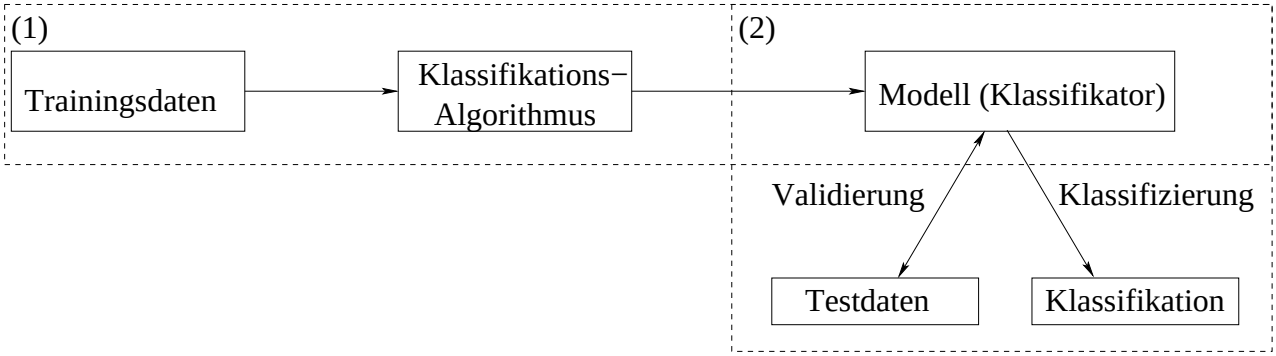


Abbildung 2.14: Prozess der Klassifizierung

Zu Beginn wird die komplette Datenmenge, d. h. bereits klassifizierte Daten, in eine Trainingsdatenmenge und eine Testdatenmenge, häufig im Verhältnis 2 : 1, aufgeteilt. Im ersten Schritt wird dann unter Verwendung der Trainingsdatenmenge ein Modell, das anschließend eine Menge von Klassifikationsbezeichnern enthält, erstellt. Mit der Trainingsdatenmenge soll ein möglichst gutes Modell erstellt werden, so dass die Anzahl der korrekt klassifizierten neuen Datensätze maximal ist. Die Qualität der Trainingsmenge ist dabei maßgeblich verantwortlich für die Qualität des Modells. Die Darstellung des Modells erfolgt bei der Klassifikation beispielsweise in Form von Klassifikationregeln, Entscheidungsbäumen oder mathematischen Formeln.

Fortsetzung Beispiel: Das Kreditinstitut benutzt zur Konstruktion des Klassifikationsmodells für Betrugsversuche zum einen vorhandene, bereits klassifizierte, Kreditkartentransaktionen und zum

anderen Informationen vom Kontoinhaber. Für die Trainingsmenge wird eine ausgewogene Anzahl an Transaktionen gewählt, wodurch sichergestellt werden soll, dass das Modell auch zukünftig die normalen Transaktionen von den Betrugsfällen gut voneinander unterscheiden kann .

Im zweiten Schritt wird das generierte Modell zur Klassifikation von neuen Datensätzen, die noch keine Klassifizierung besitzen, verwendet. Bevor das Modell jedoch zur Klassifizierung neuer Datensätze benutzt wird, sollte das Modell auf Genauigkeit überprüft werden. Eine einfache Methode zum Testen ist die Verwendung von Daten, die bereits klassifiziert sind, jedoch nicht zur Trainingsdatenmenge gehören. Methoden, die nach diesem Prinzip die Genaugkeit überprüfen sind z. B. die Confusion-Matrix (Wahrheitsmatrix) [Ora06d][WIK06], ROC (Receiver Operating Characteristics) [ROC06][Ora06d] und Lift [Ora06d], wobei die beiden letztgenannten den Nachteil haben, dass diese lediglich für eine binäre Klassifikation anwendbar sind. Eine weitere Methode ist die *Cross-Validation* [Ora06c][HK00], die zur Messung der Genauigkeit von Klassifikationsmodellen eingesetzt wird.

Im folgenden sei die `Confusion-Matrix`, die eine einfache Methode zur Überprüfung der Genauigkeit von Klassifikationsmodellen darstellt, näher erläutert. Für weitere Methoden zur Überprüfung der Genauigkeit des Klassifikators sei auf die Literatur verwiesen.

Die Confusion-Matrix ist eine $n \times n$-Matrix, dessen Zeilen den realen Werte und dessen Spalten den durch den Klassifikator vorhergesagten Werte entspricht. Die Zellen der $n \times n$-Matrix repräsentieren alle möglichen Kombinationen einer Klassifikation, wobei n die Anzahl der möglichen Kategorien bzw. Klassen ist. Der Vorteil der Confusion-Matrix ist die Beurteilung von binären und mehrwertigen Klassifikationen.

Für eine binäre Klassifikation, d. h. eine Ja/Nein-Klassifikation, ist die Confusion-Matrix eine 2×2-Matrix (siehe Abb. 2.15 a).

	Vorhersage positiv	Vorhersage negativ
real positiv	richtig positiv	falsch positiv
real negativ	falsch negativ	richtig negativ

(a) 2x2–Confusion–Matrix

	Betrug	kein Betrug
Betrug	600	30
kein Betrug	15	200

(b) Confusion–Matrix: Betrug

Abbildung 2.15: Confusion-Matrix

Die richtig vorhergesagten Fälle liegen dabei auf der Hauptdiagonalen der Confusion-Matrix, die falsch vorhergesagten in den restlichen Zellen der Matrix. In Abb. 2.15 b ist eine Confusion-Matrix für einen Klassifikator, der Betrugsfälle klassifiziert, abgebildet. Die Genauigkeit des Klassfikators läßt sich wie folgt berechnen:

a. $\text{Trefferrate} = \dfrac{\text{richtige Vorhersagen}}{\text{Gesamtzahl der Vorhersagen}} = \frac{800}{845} = 97,4\%$

b. $\text{Fehlerrate} = \dfrac{\text{falsche Vorhersagen}}{\text{Gesamtzahl der Vorhersagen}} = \frac{45}{845} = 5,3\%$

Die Confusion-Matrix gibt einen schnellen Überblick über die Güte des Klassifikators, d. h. die Klassifikation von Daten in die jeweiligen Klassen, wobei die Gewichtung von richtig--klassifizierten und falsch-klassifizierten Daten detailliert aufgeführt wird. Außerdem kann mit der Confusion-Matrix die Genauigkeit bzw. die Fehlerrate des Klassifikators gemessen werden.

Ist die Genauigkeit des Modells akzeptabel, kann das Modell zur Klassifizierung neuer Datensätze verwendet werden. Ist die Genauigkeit nicht akzeptabel, so wird oftmals von Overfitting (Überspezialisierung) oder Underfitting (Unterspezialisierung) des Modells gesprochen, d. h. das Modell wurde entweder zu genau (zu komplexes Modell) oder zu ungenau (zu einfaches Modell) an die Trainingsdaten angepasst und liefert im Vergleich schlechtes Ergebnis für unbekannte Daten. In [Mit97] wird beispielsweise Overfitting folgendermaßen definiert: *„Eine Klassenbeschreibung L ist in Bezug auf eine Menge von Trainingsdaten überspezialisiert, wenn es eine alternative Klassenbeschreibung L' gibt, so dass L in Bezug auf die Trainingsdaten eine geringere Fehlerrate hat als L', aber in Bezug auf alle möglichen Beispiele (insbesondere bisher nicht bekannte Daten wie die Testdaten) eine größere Fehlerrate hat als L'."*

Formal lässt sich das Klassifikation folgendermaßen definieren:

Definition (Klassifikation): Sei X ein Objektraum und $T \subset X$ eine sogenannte Trainingsmenge. Sei weiter $t \in T$ definiert als $t = \{A_1, ..., A_d\}$ mit $d \in \mathbb{N}$ und den Attributen $A_i, 1 \leq i \leq d$. Außerdem soll eine Zuordnung von t zu einer Klasse C_j (also $t \in C_j$) existieren, wobei $C_j \in C = \{C_1, ..., C_k\}$ mit $k \in \mathbb{N}$. Zu finden ist ein Klassifikator $K : X \rightarrow C$, der die Klassenzugehörigkeit für alle $X \setminus T$ korrekt bestimmt.

Für die Klassifikation existieren verschiedene Klassifikations-Methoden. Im Folgenden sind einige der wichtigsten und bekanntesten Methoden aufgelistet:

a.) Klassifikationsbäume (Klassifikationsregeln)
b.) Bayesian und Naive Bayesian Klassifizierung
c.) Bayesian Belief Network Klassifizierung
d.) Support Vector Machines (SVM)
e.) Weitere:
k Nearest Neighbor-Klassifizierung, Neuronale Netze, Genetische Algorithmen

a.) Klassifikationsbäume

Die Klassifizierung mit Klassifikationsbäumen, auch Entscheidungsbäume genannt, ist eine einfache und effektive Methode. In Abb. 2.16 ist ein Entscheidungsbaum abgebildet, der beispielsweise Betrugsfälle klassifiziert. Das Modell wird durch einen Baum repräsentiert, wobei der Wurzelknoten und jeder interne Knoten einem Attributtest, jeder Zweig einem Ergebnis eines Attributtests und jedes Blatt einer Klasse bzw. Klassifizierung entspricht. Neue Daten, die klassifiziert werden sollen, werden *gegen den Baum* getestet, indem die Attributwerte ausgewertet werden, d. h. beginnend beim Wurzelknoten werden die Attributwerte des Datensatzes solange sukzessive ausgewertet bis ein Blattknoten erreicht ist, also eine Klassifizierung stattgefunden hat. Die Klassifizierung kann dabei nicht nur binäre Klassifikationswerte annehmen.

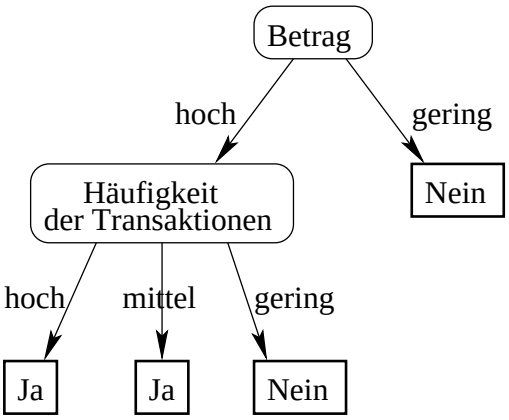


Abbildung 2.16: Modell für Betrug

Bei der Konstruktion des Klassifikationsbaums ist die Aufteilung der Trainingsmenge und die damit verbundene Aufteilungsfunktion (Aufteilungskriterium) von besonderer Bedeutung. Die Trainingsmenge wird solange in Untermengen aufgeteilt bis eine Abbruchbedingung im Konstruktionsprozess erreicht ist oder bis die Untermengen homogen sind, d. h. die Daten zu genau einem

Klassifikationswert zugeordnet sind. Das Aufteilungskriterium entscheidet wie eine Menge aufgeteilt wird, so dass die Homogenität in den entstehenden Untermengen maximal ist. In der Literatur existieren eine Menge Aufteilungskriterien, die für die Konstruktion eines Klassifikationsbaumes verwendet werden können. Im Folgenden wird auf zwei bekannte Aufteilungskriterien – Gini-Index und die Entropy – eingegangen.

Der Gini-Index und die Entropy sind Aufteilungskriterien, die die beste Aufteilung einer Menge anhand der Homogenität der entstehenden Untermengen misst. Die Homogenität einer Menge definiert sich als die Gleichheit einer Eigenschaft bzw. die Gleichartigkeit der enthaltenen Objekte, und kann durch den Grad der Verunreinigung (engl. impurity) beschrieben werden, d. h. die Störung der Gleichheit. In Abb.2.17 sind drei mögliche Aufteilungen einer Menge in zwei Untermengen abgebildet, wobei C_i die Klasse und N_1 die Untermengen beschreibt. Die Aufteilung einer Menge in zwei oder mehrere Untermengen beschreibt den Attributtest beim Klassifikationsbaum, z. B. Attribut `A<x`. Bei den Aufteilungen – von links nach rechts betrachtet – ist zu kennen, dass die entstehenden Mengen von homogen und einen niedrigen Grad an Verunreinigung nach inhomogen und einen hohen Grad an Verunreinigung wandern. Zusätzlich ist in Abb.2.17 der jeweilige Gini-Index (Berechnung siehe unten) für die Aufteilung der Menge berechnet.

Aufteilungs−bedingung	Attribut A < x	Attribut B < y	Attribut C < z
Aufteilung	<table><tr><td></td><td>N1</td><td>N2</td></tr><tr><td>C1</td><td>0</td><td>6</td></tr><tr><td>C2</td><td>6</td><td>0</td></tr></table>	<table><tr><td></td><td>N1</td><td>N2</td></tr><tr><td>C1</td><td>5</td><td>1</td></tr><tr><td>C2</td><td>1</td><td>5</td></tr></table>	<table><tr><td></td><td>N1</td><td>N2</td></tr><tr><td>C1</td><td>4</td><td>2</td></tr><tr><td>C2</td><td>3</td><td>3</td></tr></table>
Gini−Index	0.0	0.278	0.486

Abbildung 2.17: Aufteilungskriterium: Gini-Index

- Gini-Index: Der Gini[6]-Index wurde ursprünglich zur Bewertung der Einkommensverteilungen einer Volkswirtschaft entwickelt und basiert auf der Lorenzkurve [FAE06], wobei die Abweichung von der vollkommenen Gleichverteilung gemessen wird. Der Gini-Index lässt sich wie folgt berechnen:

$$\text{gini(t)} = 1 - \sum_j \left(\frac{\text{Anzahl Daten der Klasse j}}{\text{Anzahl der Daten im Knoten t}}\right)^2$$

  Zur Bewertung der Aufteilung dient die folgende Berechnung, wobei t der Knoten des Klassifikationsbaums, der in k Partitionen (spätere Nachfolgerknoten von t im Klassifikationsbaum) aufgeteilt werden soll, n_i die Anzahl der Daten in Knoten i und n die Gesamtzahl der Daten im Knoten t ist:

$$\text{gini\_split} = \sum_{i=1}^{k} \frac{n_i}{n} \times \text{gini(i)}$$

  Der Gini-Index kann auf diese Weise für jede Aufteilung berechnet werden, wobei die endgültige Aufteilung gemacht wird, wenn der Gini-Index minimal ist.
- Entropy: Die Entropy stammt aus der Informationstheorie wurde ursprünglich zur Nachrichtenübertragung verwendet. Die Zeichen in den Nachrichten haben einen unterschiedlichen Informationsgehalt in Abhängigkeit von ihrer Auftrittswahrscheinlichkeit. Ein Zeichen mit

[6]Corrado Gini (1884-1965), ital. Statistiker

geringem Vorkommen hat beispielsweise einen höheren Informationsgehalt als mit einem hohen Vorkommen. Die Entropy lässt sich wie folgt berechnen:

$$\text{entropy(t)} = -\sum_j \left(\frac{\text{Anzahl der Daten der Klasse j}}{\text{Anzahl der Daten in Knoten t}}\right) \log_2 \left(\frac{\text{Anzahl der Daten der Klasse j}}{\text{Anzahl der Daten in Knoten t}}\right)$$

Zur Bewertung der Aufteilung dient die folgende Berechnung, wobei gilt: p ist Vorgängerknoten im Klassifikationsbaum und k beschreibt die Menge der Knoten die p partitionieren sollen. Weiter gilt n_i ist die Anzahl der Daten im Knoten i und n die Gesamtzahl der Daten im Knoten p:

$$\text{Gain\_Split} = entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} \times \text{entropy(i)}\right)$$

Der Fakt, dass der Gain_Split überwiegend Aufteilungen mit vielen kleinen Partitionen bevorzugt, soll durch das Split_Info verhindert werden.

$$\text{Gain\_Ratio} = \frac{\text{Gain\_Split}}{\text{Split\_Info}} \quad \text{mit Spiel\_Info} = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Für die Konstruktion des Entscheidungsbaums (siehe [HK00]) wird, wie bei der Klassifikation üblich, eine Trainingsmenge benutzt. Anschließend wird der Entscheidungsbaum, d. h. das Klassifikationsmodell, durch Pruning[7] optimiert, indem Ausreißer (Outlier) und störende Daten (Noise) entfernt werden. Aus dem Entscheidungsbaum können ebenfalls Klassifikationsregeln der Form `IF-THEN` extrahiert werden. Diese Regeln werden für jeden Pfad des Entscheidungsbaumes vom Wurzelknoten zum Blattknoten generiert.

b.)Bayessche Klassifizierung

Die Bayessche Klassifikation ist eine statistische Klassifikation [HSC91], die die Klassifizierung von Daten anhand von Wahrscheinlichkeiten vornimmt. Dabei werden alle Attribute $A_1, \ldots, A_n$ und Klassen C_i mit $i \in \mathbb{N}$ als Zufallsvariablen angenommen. Diese Klassifikationsmethode ist im Vergleich zu anderen Methoden schnell und effizient, auch für große Datenmengen [HK00]. Die Klassifikation beruht auf dem Theorem von Bayes:

$P(A|B) = \frac{P(B|A)*P(B)}{P(A)}$, wobei $P(A)$ die Wahrscheinlichkeit von A und $P(A|B)$ die bedingte Wahrscheinlichkeit von A unter B ist.

Die Klassifikation bekommt als Eingabe eine Menge von Attributen $A_1, \ldots, A_n$ und der Bayessche Klassifikator ermittelt die Klasse C_i zu der der neue Datensatz zugehörig ist. Das Ziel ist den Ausdruck $P(C_i|A_1, \ldots, A_n)$ zu maximieren, was nach Bayes äquivalent zur Maximierung des Ausdrucks $P(A_1, \ldots, A_n|C_i) * P(C)$ ist.

Die Naive Bayes-Klassifizierung ist der normalen Bayessche Klassifizierung ähnlich. Es wird lediglich angenommen, dass die Attribute $A_1, \ldots, A_n$ unabhängig voneinander sind, und dadurch das Maximierungsproblem wie folgt vereinfacht wird:

$$P(A_1, \ldots, A_n|C_i) = P(A_1|C) \cdot \ldots \cdot P(A_n|C) \Rightarrow \text{Maximierungsproblem: } P(C) * \prod_1^n P(A_i|C)$$

Die Naive Bayessche Klassifikation für diskrete Attribute wird wie folgt definiert, wobei die Wahrscheinlichkeiten durch Häufigkeiten approximiert werden:

[7] Das Pruning bezeichnet den Vorgang der nachträglichen Vereinfachung einer gelernten Hypothese, um ein Overfitting zu verhindern.[WIK06]

$P(A_i|C_k) = \frac{|A_{ik}|}{N_k}$, wobei $|A_{ik}|$ die Anzahl Objekte ist, die das Attribute A_i besitzen und zur Klasse C_k gehören, und N_k die Anzahl der Objekte ist, die zur Klasse C_k gehören.

$P(C_k) = \frac{N_k}{N}$, N_k ist die Anzahl der Objekte der Klasse C_k und N alle Objekte.

Fortsetzung Beispiel: Das Kreditinstitut möchte mit Hilfe des Modells eine neue Transaktion klassifizieren. Folgende Informationen stehen zur Verfügung:

X = (Betrag = 'mittel', Häufigkeit der Transaktionen = 'hoch', Tageszeit = 'morgens')

Dazu müssen zunächst alle notwendigen Informationen berechnet werden, d. h. $P(X|C_i) * P(C_i)$ für $i \in \{1, 2\}$. Also alle Wahrscheinlichkeiten $P(C_i)$ und alle bedingten Wahrscheinlichkeiten $P(X|C_i)$:

P(Betrug = 'Ja') = $\frac{75}{100} = 0.75$ und P(Betrug = 'Nein') = $\frac{25}{100} = 0.25$

P(Betrag = 'mittel' | Betrug = 'ja') = $\frac{20}{75} = 0.2\bar{6}$
P(Betrag = 'mittel' | Betrug = 'nein') = $\frac{5}{25} = 0.2$
P(Häufigkeit der Transaktionen = 'hoch' | Betrug = 'ja') = $\frac{50}{75} = 0.\bar{6}$
P(Häufigkeit der Transaktionen = 'hoch' | Betrug = 'nein') = $\frac{10}{25} = 0.4$
P(Tageszeit = 'morgens' | Betrug = 'ja') = $\frac{25}{75} = 0.\bar{3}$
P(Tageszeit = 'morgens' | Betrug = 'nein') = $\frac{18}{25} = 0.72$

P(X | Betrug = 'ja') * P(Betrug = 'Ja') = $(0.2\bar{6} \cdot 0.\bar{6} \cdot 0.\bar{3}) \cdot 0.75 = 0.0\bar{4}$
P(X | Betrug = 'nein') * P(Betrug = 'Nein') = $(0.2 \cdot 0.4 \cdot 0.72) \cdot 0.25 = 0.0144$

Unter Verwendung der berechneten Wahrscheinlichkeiten ergibt sich, dass die Wahrscheinlichkeit für Betrug ($P(Betrug =' ja') = 0.0\bar{4}$) größer als für kein Betrug ($P(Betrug =' Nein') = 0.0144$) ist. Die Transaktion X wird aus diesem Grund als ein Betrugsfall klassifiziert.

c) Klassifizierung durch Bayesian Belief Networks

Das Bayessche Netz (Bayesian Belief Network) ist eine Methode aus dem Bereich der Künstlichen Intelligenz. Darunter ist ein gerichteter azyklischer Graph zu verstehen, mit dem Entscheidungen (Klassifizierung) anhand von Wahrscheinlichkeiten getroffen werden.

Die Knoten des Graphen sind Zufallsvariablen und haben zwei oder mehr mögliche Werte, die jeweils Wahrscheinlichkeiten besitzen. Der einfachste Fall sind die zwei booleschen Werte true und false mit jeweils der Wahrscheinlichkeit von 50%. Die Einteilung von Intensitäten von Erdbeben könnte z. B. die Werte *kein*, *leicht*, *mittel* und *hoch* sein.

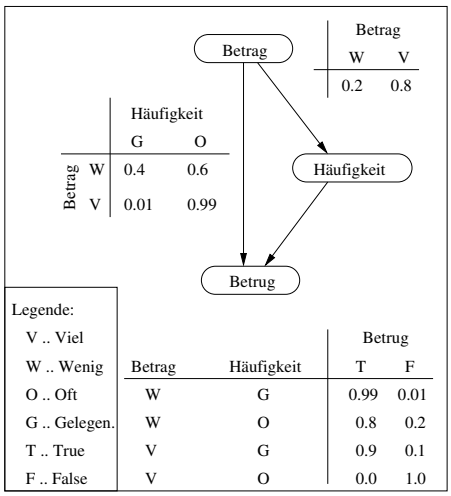


Abbildung 2.18: Bayessches Netz

Die Kanten des Graphen entsprechen statistischen Abhängigkeitsbeziehungen zwischen den Zufallsvariablen, die durch bedingte Wahrscheinlichkeiten ausgedrückt werden. In Abb. 2.18 sind die Knoten *Betrag* und *Häufigkeit (der Transaktionen)* durch eine Kante verbunden, d. h. der *Häufigkeit* ist direkt abhängig vom Knoten *Betrag* und die dazugehörigen bedingten Wahrscheinlichkeiten sind in der nebenstehenden Tabelle notiert.

Diese Abhängigkeitsbeziehungen werden zur Berechnung der Wahrscheinlichkeitsverteilung unter den Knoten des Graphen verwendet, unter Benutzung des Theorems von Bayes $P(X_1, \ldots, X_n) = \prod_{i=1}^{n}(X_i|\text{Vorgängerknoten}(X_i))$. Zur vollständigen Berechnung der Wahrscheinlichkeitsverteilung

des Bayessche Netzes sind demnach die Wahrscheinlichkeiten der Wurzelknoten (Knoten ohne Vorgängerknoten) und die bedingten Wahrscheinlichkeiten aller anderen Knoten (in Bezug auf die direkten Vorgängerknoten) notwendig. Damit ist es dann möglich mit dem Bayessche Netz bedingte Wahrscheinlichkeiten der Knoten, unter der Annahme das einige Werte bekannt sind, zu berechnen und somit eine Klassifikation zu ermöglichen. Z. B. Unter der Annahme, dass der Wert für *Häufigkeit* = `G (elegentlich)` ist, ändern sich die Wahrscheinlichkeiten der anderen Knoten des Graphen und die Wahrscheinlichkeit, dass es sich um einen *Betrug* handelt, ändert sich von `'Betrug'=true` = 45% auf `'Betrug'=true` = 90%.

d) Klassifizierung durch Support Vector Machines

Support Vector Machines (SVMs) ist eine Klassifikationsmethode, die in den letzten Jahren vielfach in der Literatur aufgekommen ist. Die Klassifikation mit dem Support Vector Machines-Algorithmus gestaltet sich recht flexibel und ist in der Genauigkeit der Klassifikation den anderen Verfahren überlegen, wodurch er sich gut zur Lösung realer komplexer Klassifikationsprobleme eignet [BM05]. Neben der reinen Klassifikation eignen sich SVMs darüber hinaus auch zur Regression. SVMs verwenden dabei das Konzept der Kernel-Modell-Substitution [Vap95]. Ein Kernel ist ein Maß, das zur Messung der Ähnlichkeit von Daten verwendet wird und der lineare Kernel stellt dabei das einfachste Modell dar.

Die Grundidee von SVM besteht in der Annahme, dass Daten die dicht an einer Entscheidungsgrenze für eine Klassifikation liegen, die sogenannten Support Vektoren, dazu benutzt werden können, um diese Grenze zu definieren (Abb. 2.19 (a)). Der SVM-Algorithmus versucht diese Daten, die dicht an einer Entscheidungsgrenze liegen zu identifizieren, so dass ein optimaler und maximaler Bereich zwischen den zu trennenden Daten entsteht und damit eine möglichst gute Verallgemeinerung für neue unklassifizierte Daten bietet.

Support Vector Maschines lassen sich zur Lösung von linearen (einfacher Fall) und nicht-linearen (schwieriger Fall) Klassifikationsproblemen einsetzen, wobei das Lösen von komplexen nicht-linearen Problemen die eigentliche Herausforderung der Klassifikation darstellt, da kein geeignetes mathematisches Modell zur Beschreibung existiert. Um es zu beschreiben, wird ein nicht-lineares Problem auf ein lineares Problem abgebildet. Im folgenden wird die Funktionsweise der SVMs anhand der unterschiedlichen Fälle erläutert.

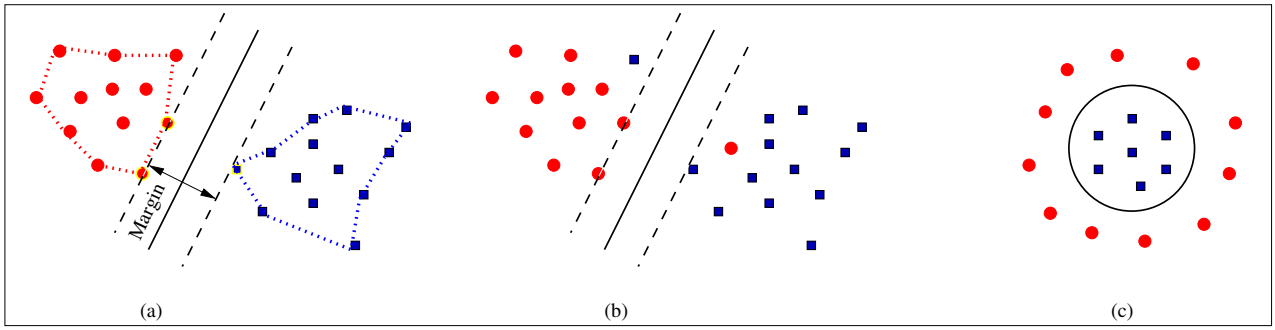


Abbildung 2.19: Darstellung der drei Fälle bei Support Vector Machines [BC00]

Linearer Fall: Gegeben sei ein binäres Klassifikationsproblem, z. B. die Identifikation von Betrugsfällen, (Abb. 2.19 (a)), d. h. eine Menge von Objekten gehört entweder zur Klasse A (Betrug) oder zur Klasse B (kein Betrug). SVM versucht die Menge mittels einer Hyper-Ebene derart zu trennen, dass die Breite (Margin) der Hyper-Ebene maximal wird. Dadurch wird erreicht, dass die Klassifizierung der Mengen weitestgehend verallgemeinert und ein Overfitting der Mengen verhindert wird. Die

Hyper-Ebene, die das Modell oder den Klassifikator darstellt, muss bevor es zur Klassifikation benutzt werden kann, erstellt werden. Zur Erstellung des Modells dient, wie bei der Klassifikation üblich, eine Trainingsmenge. Dazu sei die folgende vereinfachte Vorgehensweise betrachtet [BC00]:

- Berechnung der konvexen Hülle der betrachtenden Mengen.
- Ermittlung derjenigen Objekte in den konvexen Hüllen, die den kürzesten Abstand zwischen den Mengen haben.
- Alternativ zum vorigen Punkt kann auch versucht werden die Margin (siehe Abb. 2.19 (a)) der Hyper-Ebene zu maximieren, indem zwei unterstützende Hyper-Ebenen so lange auseinander gedrückt werden bis diese auf eine kleine Menge von Objekten, den sogenannten Support Vectors (siehe Abb. 2.19 (a) gelb-umkreiste Objekte), stoßen.
- Die Hyper-Ebene ist der Klassifikator und kann zur Klassifikation von weiteren Objekte verwendet werden.

Anmerkung: Das Problem der „Ermittlung des kürzesten Abstands zwischen den konvexen Hüllen“ ist äquivalent zur „Maximierung der Margin zwischen den unterstützenden Hyper-Ebenen“.

Linear nicht-separierbarer Fall: Gegeben sei wieder ein binäres Klassifikationsproblem, z. B. die Identifikation von Betrugsfällen, jedoch ist diesmal die Menge nicht durch eine lineare Hyper-Ebene in zwei disjunkte Teile teilbar. Der Grund dafür ist, dass sich die konvexen Hüllen der Mengen schneiden und der bisherige Lösungsweg somit ausgeschlossen ist. Für diesen Fall müssen Fehler bei der Erstellung der Hyper-Ebene zugelassen werden. Eine Möglichkeit besteht in der Reduzierung der konvexen Hüllen (Abb. 2.19 (a) und (b)) um einen definierten Grenzwert, wodurch erreicht wird, dass der Einfluss der einzelnen Objekte in der Menge reduziert wird. Durch diese Maßnahme lässt sich der bisherige Lösungsansatz wieder anwenden. Es soll ebenfalls wieder eine Maximierung der Margin und zusätzlich eine Minimierung der Fehler erreicht werden.

Nicht-linearer Fall: Es sei ein binäres Klassifikationsproblem (Identifikation von Betrugsfällen) gegeben, dass nicht gut durch ein lineares Modell beschrieben werden kann, z. B. wird ein quadratisches Modell zur optimalen Lösung benötigt. Ein nicht-linearer Fall wird aus diesem Grund durch eine nicht-lineare Abbildung $\theta : \overrightarrow{x} \longmapsto \theta(\overrightarrow{x})$ transformiert (Abb. 2.20). Dadurch wird der nicht-lineare Fall zum linearen Fall und kann wie bisher behandelt werden. Es ergibt sich zunächst das Problem eine geeignete Repräsentation als nicht-lineare Transformation zu finden. Dafür stehen verschiedene Transformationen, sogenannte Kernel, zur Verfügung, z. B. Polynomial-Kernel oder Gauss-Kernel.

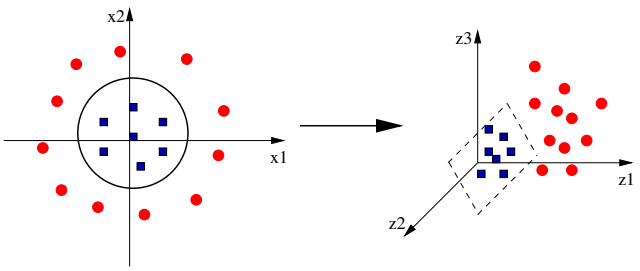


Abbildung 2.20: Behandlung Nicht-linearer Fall

2.6.5 Regression

Die Regression ist vergleichbar zur Klassifikation. Beide Verfahren stellen ein Vorhersage-Modell dar, wobei die Klassifikation benutzt wird, um diskrete und nominelle Werte vorherzusagen und Regression benutzt wird, um numerische kontinuierliche und geordnete Werte vorherzusagen.[8] Die Regression lehnt sich dabei stark an Techniken aus der Statistik an. Bei der Regression werden sogenannte Regressionsfunktionen, die allgemein die Form $y = f(x_1, \ldots x_n)$ haben, verwendet, um Beziehungen zwischen einer abhängigen Variablen und ein oder mehreren unabhängigen Variablen zu bestimmen. Die abhängige Variable soll dabei bestimmt werden, und die unabhängigen Variablen stützen die Vorhersage.

[8] Durch Diskretisierung der kontinuierlichen und geordneten Werte kann eine Regression zu einer Klassifikation umgewandelt werden.

Statistische Methoden zur Regression:

Für die Regression gibt es drei häufig verwendete statistische Ansätze [HK00] zur Lösung von Problemen, die im Folgenden benannt sind:

- **Lineare Regression:** $Y = \alpha + \beta \times X$
- **Multiple Regression:** $Y = \alpha + \beta_1 \times X_1 + \beta_2 \times X_2$
- **Nicht-Lineare Regression:** $Y = \alpha + \beta_1 \times X + \beta_2 \times X^2 + \beta_3 \times X^3 + \ldots$

Beispiel für eine Lineare Regression: Der Turm von Pisa ist ein architektonisches Wunder. Darum wurden einige Messungen durchgeführt, um die Stabilität in den kommenden Jahren vorhersagen zu können [Mar05]. Die erfassten Daten sind in der nachfolgenden Tabelle zusammengefasst und in Abb. 2.21 abgebildet. Die Neigung repräsentiert den Abstand eines auf den Boden projizierten Punktes der Spitze zur Basis der Turms.

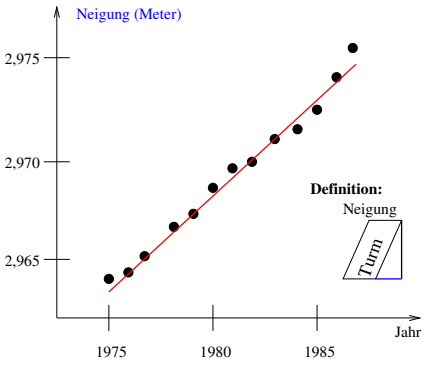


Abbildung 2.21: Beispiel: Pisa

Aus dem Diagramm 2.21 ist zu entnehmen, dass es eine lineare Abhängigkeit zwischen dem Jahr und der Neigung gibt. Die Stabilität bzw. die Neigung des Turm lässt sich also anhand der linearen Regression (rote Gerade in Abb. 2.21) für die zukünftigen Jahre bestimmen.

Jahr	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984
Neigung	2,9642	2,9644	2,9656	2,9667	2,9673	2,9688	2,9696	2,9698	2,9713	2,9717

Epsilon Support Vector Machine-Regression:

Die Support Vector Machines können ebenfalls zur Regression, d. h. zur Vorhersage von numerische kontinuierlichen Werten, benutzt werden [BC00] [SS98]. Der Support Vector Machine-Algorithmus soll bei der Regression jedoch abweichend von der Verwendung bei der Klassifikation keine Klassen durch Hyper-Ebenen separieren, sondern soll versuchen die Daten durch eine Hyper-Ebene miteinander zu verbinden.

Bei der Regression mit Support Vector Machines soll, wie bei den anderen Regressionsproblemen auch, eine Regressionsfunktion `f`, die eine funktionale Abhängigkeit zwischen ein oder mehreren Variablen darstellt, ermittelt werden.

Die Support Vector Machines für Regression verwendet zur Bestimmung der Regressionsfunktion `f` eine ϵ-Umgebung um die Funktion `f` herum. In Abb. 2.22 ist eine lineare Regressionsfunktion und die ϵ-Umgebung (gestrichelte Linien) dargestellt. Alle Funktionswerte, die sich in dieser ϵ-Umgebung der Regressionsfunktion befinden, werden nicht als Fehler sondern als korrekte Funktionswerte der Funktion `f` betrachtet.

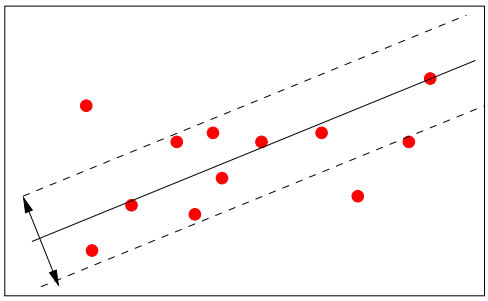


Abbildung 2.22: ϵ-SVM Regression

Es soll demnach mit dem Support Vector Machines-Algorithmus versucht werden, eine Regressionsfunktion zu ermitteln, in der die Anzahl der Werte innerhalb dieser ϵ-Umgebung maximal und der Wert für ϵ minimal ist. Zur Bestimmung werden, wie bei der Klassifikation, verschiedene Kernel, z. B. Gauss-Kernel, und eine Trainingsmenge verwendet, so dass mit der ermittelten Regressionsfunktion neue Werte korrekt vorhergesagt werden können. Dabei kann der Wert für Epsilon, also die maximale Abweichung vom beobachteten Wert der Trainingsmenge und Funktionswert von `f`, sowohl automatisch berechnet als auch manuell vorgegeben werden [BC00].

2.7 Interpretation

Damit die Suche nach implizitem Wissen, das durch den KDD-Prozess und dem Data Mining gefunden wurde, zu einem Ergebnis kommt, muss das entdeckte Wissen in irgendeiner Form dem Benutzer präsentiert werden, so dass dieser eine Interpretation vornehmen kann. Dabei hat sich gezeigt, dass die grafische Visualisierung von Wissen eine wirksame Methode darstellt, um den Benutzer zu unterstützen, die zum Teil komplexen Muster und das damit verbundene Wissen zu meistern. Eine Visualisierung wird also benötigt, um das Wissen besser verstehen und im weiteren Prozess verarbeiten zu können.

Der Benutzer muss also in der Lage sein, anhand der Visualisierung der gefundenen Ergebnisse, adäquate Interpretationen durchführen zu können. Dabei ist zu beachten, dass die Visualisierungstechniken unterschiedlich gut Wissen darstellen können, beispielsweise muss zur Visualisierung von Ergebnissen des Clustering eine andere Technik als für Ergebnisse der Assoziationsanalyse herangezogen werden. Neben der reinen Visualisierung der Ergebnisse, stellt sich zunächst die Frage nach der Absicht, d. h. welches Ziel sollte ursprünglich mit der Data Mining-Methode verfolgt werden. Daraus ergeben sich die in Abb. 2.23 dargestellten Interpretationsrichtungen.

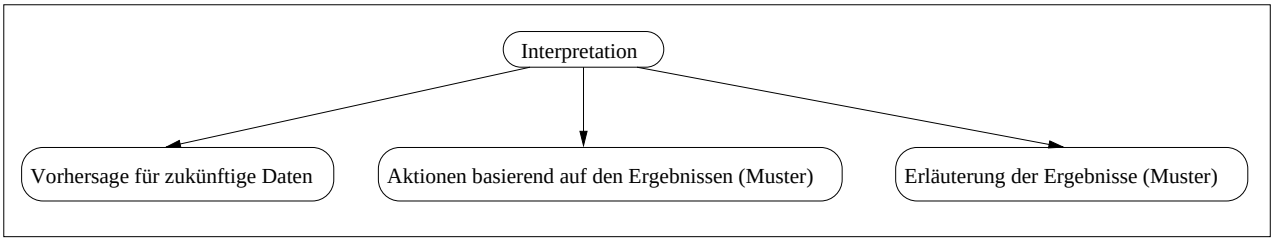


Abbildung 2.23: Richtungen der Interpretation

Als einfache Visualisierungsmöglichkeiten bieten sich die nachfolgenden Techniken an:

- Regeln (Assoziationsregeln 2.6.3 und Klassifikationsregeln 2.6.4)
- Entscheidungsbäume 2.6.4, Netze (Bayesian Netze und Neuronale Netze)
- Tabellen
- Diagramm (1D-/2D-/3D-Diagramm, Kuchendiagramm, Balkendiagramm)
  n-dimensionale Diagramme für $n \in \{1, 2\}$ haben den Vorteil, dass darin abgebildete Daten einfach verständlich sind. Das Problem ist jedoch, dass wichtige Zusammenhänge häufig zwischen Daten mit $n \geq 3$ zu finden sind und eine komplexere Visualisierung zur Folge haben. Bei höher-dimensionalen Daten wird daher versucht, diese auf weniger Dimensionen abzubilden. Das hat jedoch zur Folge, dass die Darstellungen nicht mehr intuitiv interpretierbar sind. In der Astronomie beispielsweise werden Himmelskörper anhand von mehreren Merkmalen geclustert [FSM92], um neue Himmelskörper zu identifizieren. Aus den 40 Attributen, die die Himmelskörper beschreiben, sind durch acht Attribute für das Clustering interessant. Eine sinnvolle Visualisierung der 8 Attribute ist hierbei kaum möglich.

Neben der Intention das implizite Wissen richtig interpretieren zu können, ist die Sensibilisierung des Knowledge Discovery Prozesses von Interesse. Die Erkenntnisse und Ergebnisse, die bisher aus dem Knowledge Discovery Prozess extrahiert wurden, sollen dazu verwendet werden, um den Prozess weiter zu steuern und zu sensibilisieren, damit noch weiteres implizites Wissen aufgedeckt werden kann.

3 Oracle Knowledge Discovery und Data Mining

In diesem Kapitel soll die Data Mining-Cartridge von Oracle beschrieben werden. Zunächst wird ein kurzer Überblick gegeben, wobei auf die Ziele und Intentionen von Oracle eingegangen wird. Danach werden die Möglichkeiten für die Benutzung, der in der Cartridge zur Verfügung gestellten Funktionalitäten, beschrieben. Anschließend befasst sich dieses Kapitel mit dem Aspekt, welche der in Kapitel 2 vorgestellten Techniken in der Data Mining Cartridge letztendlich umgesetzt worden sind, und wie diese eingesetzt werden können, um damit Data Mining zu betreiben. Dieses Kapitel soll jedoch keine vollständige Beschreibung oder ein Benutzerhandbuch der Oracle Data Mining sein, sondern soll einen Einblick geben, inwieweit Oracle Methoden für das Data Mining und allgemein für den KDD-Prozess bereitstellt. Des weiteren soll ein Verständnis über die Funktionsweisen und darin enthaltenen Beschränkungen der implementierten Methoden gegeben werden.

3.1 Überblick

Oracle hat in seinem Datenbanksystem Oracle Database 10g R2 den Funktionsumfang um Data Mining-Funktionalitäten auf einen stabilen Stand erweitert, so dass eine breite Basis für das Data Mining zur Verfügung steht. Die Erweiterung wird Oracle Data Mining (ODM) genannt und beinhaltet Funktionen und Algorithmen, die zur Entwicklung und zum Betrieb von Data Mining-Anwendungen benutzt werden können. Dabei wird der Begriff des Data Mining in Oracle synonym mit dem Begriff der Knowledge Discovery in Databases aus Kapitel 2 benutzt. Der Knowledge Discovery-Prozess, der sich wie in Kapitel 2.1 beschrieben, in sechs Teilschritte unterteilt, wird dabei von ODM in den Schritten **Preprocessing**, **Transformation** und **Data Mining** unterstützt. Abb. 3.1 soll einen groben Überblick über die Methoden und Techniken in der jeweiligen Phase geben, die die Cartridge zur Verfügung stellt. In den folgenden Abschnitten soll nun die Unterstützung von ODM in den einzelnen Schritten des KDD-Prozesses näher betrachtet und deren Funktionsweisen und Beschränkungen jeweils kurz beschrieben und demonstriert werden.

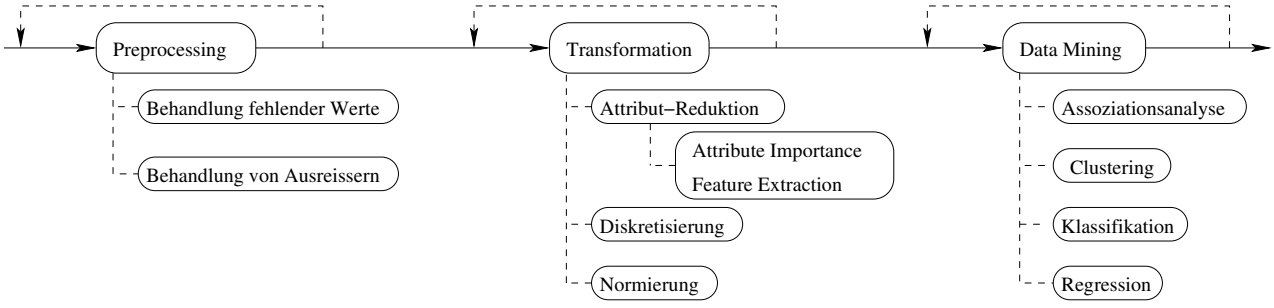
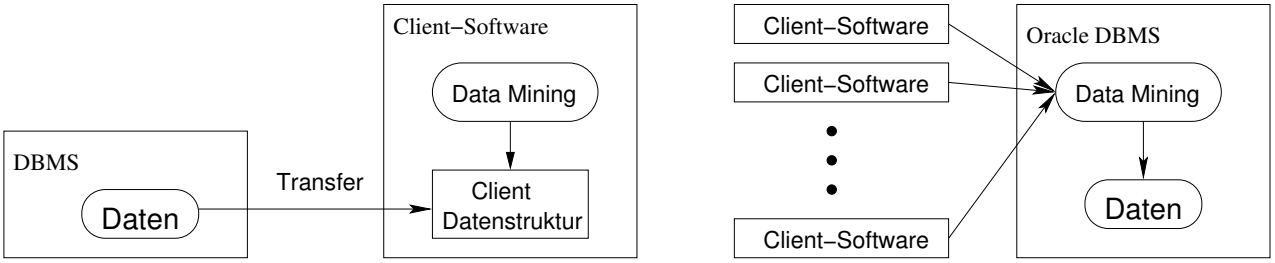


Abbildung 3.1: Knowledge Discovery in Oracle

Zunächst werden jedoch die Möglichkeiten die Data Mining-Cartridge zu benutzen, die grundlegenden Voraussetzungen für die Benutzung und das allgemeine Prinzip – Organisation und Management – von ODM vorgestellt.

3.2 Programmable Interface

Oracle bietet durch die Erweiterung, um die Data Mining-Cartridge, die Möglichkeit, das Data Mining vollständig in der Datenbank ablaufen zu lassen, wodurch alle Operationen auf objekt-relationalen Tabellen oder Sichten durchgeführt werden. Dies hat im Vergleich zu früheren Lösungen den Vorteil, dass die Daten nicht zwischen der Datenbank und einer Client-Software transferiert werden müssen. Außerdem wird eine effizientere Möglichkeit der Analyse, des Datenmanagements und der Datensicherheit gewährleistet. In Abb. 3.2 ist der Aspekt der Integration der Data Mining-Funktionalität abgebildet.



(a) Data Mining ohne DBMS–Unterstuetzung (b) Oracle Data Mining

Abbildung 3.2: Data Mining mit und ohne DBMS-Unterstützung

Dementgegen besteht ein Nachteil zu den Lösungen, in denen die Data Mining-Funktionalität in der Client-Software integriert ist, darin, dass die Erweiterbarkeit der Data Mining-Methoden beschränkt ist. Durch den proprietären Charakter der Data Mining-Cartridge von Oracle wird eine Erweiterung um neue Funktionalitäten oder Algorithmen nicht unterstützt[ORA06a].

Der Zugriff auf die Data Mining-Funktionalität erfolgt entweder über eine PL/SQL-Schnittstelle oder über eine Java-Schnittstelle (Programmable Interface). Dabei ist zu bemerken, dass die Java-Schnittstelle auf der PL/SQL-Schnittstelle basiert, wodurch die PL/SQL-Schnittstelle mindestens so viel Funktionalität wie die Java-Schnittstelle bietet. Das Data Mining wird somit vollständig aus der Client-Software entfernt, und nur noch die Eingabe und die Ausgabe, d. h. das Starten des KDD-Prozesses und die Visualisierung der Ergebnisse obliegt dem Client.

Die beiden Programmable Interfaces sind dabei derart konzipiert, dass diese vollständig kompatibel zueinander sind. Die Interoperabilität zwischen den beiden Programmable Interfaces ist dadurch gewährleistet. Es kann also beispielsweise das Preprocessing und die Transformation der Daten über die Java-API erfolgen, das Data Mining über die PL/SQL-API durchgeführt werden und anschließend die Ergebnisse wiederum über die Java-API abgerufen und verarbeitet werden, ohne dass es zu einer Inkompatibilität kommt[Ora06b].

3.2.1 PL/SQL-Schnittstelle

Die PL/SQL-Schnittstelle bietet die Möglichkeit, die Data Mining-Funktionalität in PL/SQL-Programme zu integrieren. Die Schnittstelle für das Data Mining teilt sich in zwei Packages auf:

- `DBMS_DATA_MINING`-Package
- `DBMS_DATA_MINING_TRANSFORM`-Package

Das `DBMS_DATA_MINING`-Package umfasst die gesamte Funktionalität, die sich mit dem Data Mining-Schritt des KDD-Prozesses befasst. Außerdem werden Funktionen bereitgestellt, mit denen die

Ergebnisse des Data Mining abgerufen und eingesehen werden können. Und das DBMS_DATA_MINING_TRANSFORM-Package umfasst alle Funktionen, mit denen das Preprocessing und die Transformation der Daten durchgeführt werden können, bevor die Daten mit den Funktionen im DBMS_DATA_MINING-Package verarbeitet werden. Die Funktionen in diesen beiden Packages werden im weiteren Verlauf in den Abschnitten über Preprocessing, Transformation und Data Mining beschrieben. Für eine detaillierte Beschreibung der Methoden bzgl. deren Parameter sei jedoch auf die Literatur [Ora06f][Ora06h] verwiesen.

3.2.2 Java-Schnittstelle

Die Java Data Mining-Schnittstelle (JDM-Schnittstelle) ist eine von Oracle definierte Spezifikation für das Aufgabengebiet des Data Mining. Die Schnittstelle bietet die Möglichkeit die Data Mining-Funktionalität in Java-Anwendungen zu integrieren. Es wird dabei eine Unterstützung zur Erstellung, Speicherung, Zugriff und Wartung von Daten und Metadaten bezüglich des Data Mining, der Data Mining-Ergebnissen und der Datentransformationen gegeben. Eine ausführliche Anleitung zur Benutzung der JDM-Schnittstelle ist in [Ora06b] Kapitel 6 und 7 zu finden.

3.3 Voraussetzungen

Das Data Mining mit der Oracle Data Mining-Cartridge unterliegt Bedingungen, die erfüllt sein müssen, damit die Methoden und Algorithmen funktionieren können. Sind die jeweiligen Bedingungen, auch Voraussetzungen genannt, nicht erfüllt, so funktioniert die Methode oder Funktion nicht bzw. quittiert mit einer entsprechenden Fehlermeldung.

Für die Eingabe der Data Mining-Funktionen werden relationale Tabellen oder Sichten erwartet, wobei jede Zeile der Tabelle (bzw. Sicht) ein einzelnes Objekt, und jedes Attribut eine Eigenschaft des Objektes darstellt. Die Attribute dürfen dabei nur einfache Datentypen, z. B. NUMBER oder VARCHAR2, annehmen. Jedes Objekt ist darüber hinaus eindeutig über einen Primär- oder Unique--Schlüssel zu identifizieren. Dieser Primär- bzw. Unique-Schlüssel wird anschließend in Form eines Parameters jeweils der Data Mining-Methode übergeben (siehe Abschnitt 3.4 – Listing 3.2).

Die Beschränkung, das jedes Objekt durch einen Primär- bzw. Unique-Schlüssel identifizierbar sein muss, ist jedoch relativ restriktiv in Bezug auf das Data Mining, da dadurch zu keinem Objekt mehrwertige Beziehungen möglich sind (z. B. beim Warenkorb, wo mehrere Waren einem Einkauf zugeordnet sind). Darum bietet Oracle die Möglichkeit, mehrwertige Beziehungen (1:N-Beziehungen) durch die Verwendung von geschachtelten Tabellen (*nested tables*) zu realisieren.

ODM benutzt für mehrwertige Beziehungen die Datentypen DM_NESTED_NUMERICALS für numerische Werte und DM_NESTED_CATEGORICAL für kategorische Werte [Ora06h]. Diese Datentypen repräsentieren Mengen von Attribut-Wert-Paaren, die genau einem Objekt zugeordnet sind. Durch die geschachtelten Tabellen ist es möglich komplexe Beziehungen in Daten darzustellen. Es lassen sich somit beispielsweise Daten in Form von Transaktionen (*transactional data*), wie die bei der Warenkorb-Analyse (d. h. zu einem Warenkorb gehören n unterschiedliche Produkte), darstellen und mit ODM verarbeiten.

Eine weitere Beschränkung liegt in der Anzahl der Attribute, die verarbeitet werden können, dessen Grenze bei 1000 Attributen liegt. Um diese Beschränkung teilweise zu umgehen, bietet ODM wiederum die Verwendung von den geschachtelten Tabellen (DM_NESTED_NUMERICALS und DM_NESTED_CATEGORICAL) an, wenn die Anzahl der Attribute einer Tabelle respektive Sicht mehr als

1000 übersteigt [Ora06e]. Dadurch werden mehrere Attribute zu einem einzigen Attribut zusammengefasst und somit die Anzahl der Attribute reduziert.

Zusammenfassend lässt sich sagen, dass die folgenden Voraussetzungen mindestens erfüllt sein müssen, damit die Funktionen und Algorithmen in der Data Mining-Cartridge die Daten überhaupt verarbeiten können:

- Die Daten müssen sich alle in einer einzigen Tabelle oder Sicht befinden.
- Jedes Objekt (Datensatz) in der Tabelle oder Sicht muss durch einen eindeutigen Primär- bzw. Unique-Schlüssel identifizierbar sein.
- Die Spalten der Tabelle bzw. Sicht müssen einen der folgenden Datentypen besitzen:
  - `INTEGER`, `NUMBER`, `FLOAT`
  - `VARCHAR2`, `CHAR`
  - `DM_NESTED_NUMERICALS`, `DM_NESTED_CATEGORICAL` (geschachtelte Tabellen)

Anmerkung zu den Datentypen: Obwohl ODM auch unstrukturierte Datentypen, z. B. Attribute vom Typ `TEXT`, als Eingabe akzeptiert und verarbeiten kann, wird eine Verarbeitung von Datentypen mit Bezug zu Spatial nicht unterstützt [Ora06d].

3.4 Management-Prinzip von Oracle Data Mining

Das zentrale Objekt in ODM stellt ein sogenanntes Modell (Modell-Objekt) dar. Das Modell-Objekt ist dabei vergleichbar mit einer Datenstruktur, worin die kompletten Informationen bezüglich einer Data Mining-Aufgabe gespeichert sind. Im Modell-Objekt sind z. B. Informationen wie die verwendete Data Mining-Methode, der verwendete Data Mining-Algorithmus, die benutzten Einstellungen für den Algorithmus und das Ergebnis der Data Mining-Aufgabe enthalten. Das Prinzip des Modell-Objekts in ODM bewirkt, dass für jede neue Data Mining-Aufgabe entsprechend ein neues Modell-Objekt angelegt wird.

Das Speicher-Management der Modell-Objekte übernimmt dabei das Datenbanksystem. Eine Übersicht über die vorhandenen Modell-Objekte wird über die Tabelle `DM_USER_MODELS` realisiert. Jedes Modell-Objekt wird dabei über seinen Namen eindeutig identifiziert. Des weiteren werden die Data Mining-Methode, der verwendete Algorithmus und die Größe des Modell-Objektes aufgelistet. Listing 3.1 bietet eine Beispiel-Übersicht über die erstellten Modell-Objekte, wobei jeder Benutzer seine eigenen Modell-Objekte in seinem Standard-Schema erstellt und nutzt. Die Benutzung oder Änderung von Modell-Objekten in anderen Schemata ist in ODM nicht vorgesehen.

```
SQL> select name, function_name, algorithm_name from dm_user_models

NAME                    FUNCTION_NAME       ALGORITHM_NAME
----------------------- ------------------- ----------------------
                        ...
MODEL_CLUS_KM_CARS      CLUSTERING          KMEANS
MODEL_CLASS_DT_IRIS     CLASSIFICATION      DECISION_TREE
MODEL_REGR_SVM_CARS     REGRESSION          SUPPORT_VECTOR_MACHINES
MODEL_ASSO_ITEMS        ASSOCIATION         APRIORI
                        ...
```

Listing 3.1: Überblick über Data Mining-Modelle

Konstruktion von Modell-Objekten:

Das Modell-Objekt für eine Data Mining-Aufgabe wird über die Methode `CREATE_MODEL` im Package `DBMS_DATA_MINING` erstellt. Durch den Aufruf dieser Methode wird gleichzeitig der Prozess des Data Mining durchgeführt. Es ist darüber hinaus zu beachten, dass beim Aufruf dieser Methode die gesamten Attribute der Tabelle bzw. Sicht, mit Ausnahme des Primär- bzw. Unique-Schlüssel der zur eindeutigen Identifikation der Objekte (Datensätze) dient, in den Data Mining-Prozess einbezogen werden. Der Ausschluss von Attributen ist bei der Zusammenstellung der Datentabelle oder spätestens nach dem Transformationsschritt durchzuführen.

Die Einstellungen für die Parameter, die für die Data Mining-Methode und den Data Mining-Algorithmus vom Benutzer angegeben und angepasst werden können, werden in einer sogenannten Konfigurationstabelle gespeichert. Wenn keine Konfigurationstabelle explizit angegeben wird, so wird eine interne Konfigurationstabelle mit den Standardeinstellungen der Parameter für den jeweiligen Algorithmus verwendet. Diese benutzerdefinierte Konfigurationstabelle wird beim Erzeugen des Modells mit übergeben und überschreibt dadurch die Standardeinstellungen. Die Einstellungen für die Parameter und die Standardeinstellungen werden im Folgenden bei den Data Mining-Algorithmen eingehender beschrieben. Das Schema der Konfigurationstabelle ist folgendermaßen:

Spaltenname	Datentyp	Bedeutung
setting_name	VARCHAR2(30)	Name des Parameters
setting_value	VARCHAR2(128)	Wert des Parameter

Die Methode `CREATE_MODEL` benötigt einige Parameter, die zur Konstruktion eines Modells notwendig sind. Die Parameter und deren jeweilige Bedeutung werden im Folgenden beschrieben:

1. `model_name` – Name, unter dem das Modell gespeichert wird
2. `mining_function` – Data Mining-Methode, die verwendet wird
3. `data_table_name` – Name der Tabelle, die die Daten für das Data Mining enthält
4. `case_id_column_name` – eindeutiger Identifier, der ein Objekt eindeutig identifiziert
5. `setting_table_name` – Konfigurationstabelle für die Einstellungen des Algorithmus
6. `target_column_name` – Klassifikationsattribut (`NULL` für deskriptive Modelle)

Anmerkung: Bei den Parametern der Methode `CREATE_MODEL` ist der Wertevorrat für den Parameter `mining_function` auf die folgenden Werte begrenzt: a) ASSOCIATION b) CLUSTERING c) CLASSIFICATION d) REGRESSION e) ATTRIBUTE_IMPORTANCE f) FEATURE_EXTRACTION .[1] Die übrigen Parameter der Methode `CREATE_MODEL` unterliegen keinen festen Vorgaben.

Beispiel: Das Code-Fragment in Listing 3.2 beschreibt die Konstruktion eines Modell-Objekts mit dem Namen `ASSO_MODEL` und startet damit gleichzeitig das Data Mining. Als Data Mining-Methode wird in diesem Fall die Assoziationsanalyse verwendet und durch die Einstellungen in der Konfigurationstabelle `ASSO_SETTINGS` wird der Algorithmus gesteuert. Die Einstellungen könnten die Angabe des Minimum-Supports und der Minimum-Confidence sein (siehe Kapitel 2.6.3).

```
DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'ASSO_MODEL',
    mining_function     => DBMS_DATA_MINING.ASSOCIATION,
    data_table_name     => 'data',
    case_id_column_name => 'object_id',
    setting_table_name  => 'ASSO_SETTINGS');
```

Listing 3.2: Konstruktion eines Modell-Objektes (Assoziationsanalyse)

[1] e) und f) werden in ODM in Anlehnung an [HK00] als Data Mining-Methoden betrachtet

Informationen im Modell-Objekt:

Das Modell-Objekt speichert alle relevanten Informationen zu einer Data Mining-Aufgabe. Um an diese gespeicherten Informationen zu gelangen, stellt ODM für jede Data Mining-Methode und jeden Data Mining-Algorithmus (Zugriffs-)Methoden bereit, mit denen auf das Modell-Objekt zugegriffen werden kann. Beispielsweise lassen sich damit die Ergebnisse der Assoziationsanalyse – die Assoziationsregeln und die *frequent itemsets* (siehe Kapitel 2.6.3) – abrufen und ausgeben.

Die Methoden mit denen Informationen aus dem Modell-Objekt abgerufen werden können, unterscheiden sich zum einen in Methoden, die das Ergebnis einer Data Mining-Aufgabe abrufen und verarbeiten, und zum anderen in Methoden, die die Einstellungen für eine Data Mining-Aufgabe abrufen. In der folgenden Auflistung werden diese Methoden kurz beschrieben:

- Die `GET_MODEL_DETAILS`-Methode erlaubt den Zugriff auf alle Ergebnisse, falls vorhanden, die durch das Data Mining ermittelt wurden.
- Die Methode `GET_MODEL_SIGNATURE` und `GET_MODEL_SETTINGS` liefern eine Beschreibung des Modell-Objektes, d. h. Informationen die bei der Erstellung des Modells benutzt wurden und entsprechend einen Einfluss hatten.
- Die Methode `GET_DEFAULT_SETTINGS` liefert zu einer Data Mining-Methode die Standardeinstellungen für die Parameter.

Anmerkung: Die Informationen zu einem Modell sind abhängig vom Data Mining-Algorithmus und werden dementsprechend in den nächsten Abschnitten, in denen es um die Data Mining-Methoden und Algorithmen geht, beschrieben und demonstriert.

3.5 Oracle Knowledge Discovery

Die Oracle Data Mining-Cartridge besteht aus den PL/SQL Packages `DBMS_DATA_MINING_TRANSFORM` und `DBMS_DATA_MINING`, worin die gesamten Funktionen, die für die Knowledge Discovery notwendig sind, enthalten sind. Dabei fasst Oracle die Schritte Preprocessing und Transformation der Knowledge Discovery im Package `DBMS_DATA_MINING_TRANSFORM` zusammen. Die Funktionen um das Data Mining sind im Package `DBMS_DATA_MINING` zusammengefasst.

Die Einteilung und der Umfang der Funktionen in ODM wird an [HK00] angelehnt und entspricht nicht dem in Kapitel 2 vorgestellten Data Mining. In [HK00] wird abweichend die Ausreißer-Analyse und die Attribut-Reduktion als Data Mining-Methoden beschrieben. Da es sich jedoch bei der Ausreißer-Analyse und der Attribut-Reduktion um spezielle Anwendungen der Data Mining-Methoden (Kapitel 2) handelt, wird die Einteilung nach Kapitel 2 beibehalten.

3.5.1 Oracle Preprocessing

Bevor die Daten von in Oracle bereitgestellten Data Mining-Methoden verarbeitet werden können, können diese wie in Kapitel 2.4 beschrieben, vorverarbeitet werden. ODM stellt dafür Funktionen bereit mit denen die Daten auf- und vorbereitet werden können, damit diese den geforderten Richtlinien bzw. Vorgaben der Algorithmen entsprechen oder damit die Algorithmen besser funktionieren.

Der Preprocessing-Schritt ist in ODM optional, d. h. die Data Mining-Algorithmen funktionieren, so lange die Voraussetzungen in Abschnitt 3.3 erfüllt sind, auch ohne eine vorherige Vorverarbeitung.

Für den Preprocessing-Schritt werden von ODM Funktionen zur Verfügung gestellt, die hauptsächlich im Package `DBMS_DATA_MINING_TRANSFORM` der PL/SQL-Schnittstelle enthalten sind. In

ODM werden jedoch auch spezielle Data Mining-Funktionen vorgestellt, die im Grunde in den Preprocessing-Schritt gehören, aber aufgrund des Modell-Objekt-Prinzips und dem Aspekt, das es sich dabei um eine spezielle Anwendung einer Data Mining-Methode handelt, im Package `DBMS_DATA_MINING` enthalten sind. In ODM sind folgende Preprocessing-Maßnahmen (siehe Kapitel 2.4) umgesetzt worden:

(a) Behandlung von fehlenden Werten

(b) Behandlung von Ausreißern

Allgemeines Prinzip in Oracle (Preprocessing und Transformation):

In Oracle funktioniert das Preprocessing und die Transformation (beschränkt auf die Funktionen im Package `DBMS_DATA_MINING_TRANSFORM`) nach dem folgenden Prinzip, wobei das `X` für die Art des Preprocessing respektive Transformation steht, z. B. Behandlung fehlender Werte für numerische Attribute: Funktion `CREATE_MISS_NUM` mit `X` = `MISS_NUM`.

1. Zunächst wird eine Tabelle mit der Funktion `CREATE_X` erstellt, die die Informationen in Form von Attribut-Wert-Paaren für die durchzuführende Aktion (Preprocessing bzw. Transformation) speichern soll.
2. Die Funktion `INSERT_X` berechnet anschließend die Attribut-Wert-Paare für die ausgewählten Attribute und speichert diese in der Tabelle, die durch die Funktion `CREATE_X` erstellt worden ist.
3. Mit der Funktion `XFORM_X` wird schließlich die eigentliche Aktion (Preprocessing bzw. Transformation) durchgeführt, wobei die Attribut-Wert-Paare aus der obigen Tabelle benutzt werden. Das Ergebnis wird durch eine Sicht bereitgestellt, mit der anschließend weitergearbeitet wird.

(a) Behandlung von fehlenden Werten:

Eine Behandlung der fehlenden Werte ist in Oracle Data Mining immer dann sinnvoll und angebracht, wenn der Algorithmus diese fehlinterpretieren könnte[Ora06b].

Im ersten Schritt zur Behandlung von fehlenden Werten wird die Tabelle angelegt, die die Werte zum Auffüllen der `NULL`-Werte der verschiedenen Attribute speichern soll. Es werden der Name des Attributs (`col`) und der dazugehörige Wert (`val`) des Attributs gespeichert. Diese Tabelle hat das folgende Schema:

Spaltenname	Datentyp	Bedeutung
col	VARCHAR2(30)	Name des Attributs
val	VARCHAR2(4000) bzw. NUMBER	Füllwert für `NULL`-Werte

Im zweiten Schritt zur Behandlung von fehlenden Werten werden die entsprechenden Werte für die numerischen bzw. die nicht-numerischen Attribute ermittelt. ODM stellt für numerische und für nicht-numerische Attribute jeweils eine Methode zur Verfügung (siehe Kapitel 2.4):

- **numerisch:** Berechnung des Durchschnittswerts
- **nicht-numerisch:** Wert mit dem häufigsten Vorkommen

Im dritten Schritt wird die Behandlung von fehlenden Werten abgeschlossen, indem eine Sicht auf der Datentabelle erzeugt wird. Bei der Definition der Sicht werden die ermittelten Werte für die Attribute benutzt, um damit die `NULL`-Werte der Datentabelle aufzufüllen.

In Listing 3.3 ist die Behandlung von fehlenden numerischen Werten beispielhaft dargestellt. Die Behandlung von nominalen Werte erfolgt analog, nur mit dem Unterschied, dass andere Funktionen benutzt werden.

```
BEGIN
 -- Definition der Tabelle zur Behandlung der fehlenden Werte
 -- (Tabelle speichert die Mittelwerte der Attribute, die im
 -- zweiten Schritt ermittelt werden)
 DBMS_DATA_MINING_TRANSFORM.CREATE_MISS_NUM('miss_num_def_table');

 -- Mittelwerte für die Attribute werden ermittelt
 -- Ausgenommen sind die Attribute id und column_a
 DBMS_DATA_MINING_TRANSFORM.INSERT_MISS_NUM_MEAN(
        'miss_num_def_table',
        'data_table',
        DBMS_DATA_MINING_TRANSFORM.Column_List{'id', 'column_a'}
 );

 -- Erstellt eine Sicht auf Basis der Datentabelle, in der die
 -- fehlenden Werte ersetzt wurden
 DBMS_DATA_MINING_TRANSFORM.XFORM_MISS_NUM(
        'miss_num_def_table',   -- Infos über die Mittelwerte
        'data_table',           -- Datentabelle
        'miss_num_view');       -- Ergebnis: Sicht
END; /
```

Listing 3.3: Behandlung fehlender numerischer Werte

(b) Behandlung von Ausreißern:

In ODM werden Ausreißer hauptsächlich durch die Clipping-Methode behandelt, aber auch durch Data Mining-Methoden, z. B. der *One-Class Support-Vector-Machine*-Algorithmus, können in Oracle zur Behandlung von Ausreißer-Werten (siehe Kapitel 2.4) eingesetzt werden. Eine Behandlung der Ausreißern ist in ODM notwendig, da einige Algorithmen empfindlich darauf reagieren und somit schlechtere Ergebnisse liefern könnten [Ora06d]. Im Folgenden werden diese beiden Möglichkeiten zur Behandlung von Ausreißern vorgestellt.

1.) Clipping:

Im ersten Schritt zur Behandlung von Ausreißern wird die Tabelle angelegt, die die Werte für die Grenzen und die alternativen Werte für die Ausreißer der verschiedenen Attribute speichern soll. Es werden der Name des Attributs (`col`), die untere (`lcut`) und obere (`rcut`) Grenze für den *normalen* Bereich und die alternativen Werte für die unteren (`lval`) und oberen (`rval`) Ausreißer gespeichert. Diese Tabelle hat somit das folgende Schema:

Spaltenname	Datentyp	Bedeutung
col	VARCHAR2(30)	Name des Attributs
lcut	NUMBER	untere/linke Grenze
lval	NUMBER	unterer/linker Wert
rcut	NUMBER	obere/rechte Grenze
rval	NUMBER	oberer/rechter Wert

Im zweiten Schritt zur Behandlung von Ausreißern werden die entsprechenden Werte für `lcut`, `lval`, `rcut` und `rval` für numerische Attribute ermittelt. In ODM wird Clipping durch die zwei Ansätze Winsorizing und Trimming realisiert (siehe Kapitel 2.4):

- **Winsorizing:** Behandlung der Ausreißer durch Setzen der Werte: `lval=lcut` und `rval=rcut`

- **Trimming:** Behandlung der Ausreißer durch Setzen der Werte: `lval=NULL` und `rval=NULL`

Im dritten Schritt wird die Behandlung von Ausreißern abgeschlossen, indem wiederum eine Sicht auf der Datentabelle erstellt wird, und die ermittelten Werte zur Beseitigung der Ausreißer in der Datentabelle benutzt werden.

Die Behandlung von Ausreißern erfolgt dabei nach dem Prinzip der Behandlung von fehlenden Werten (siehe Punkt *(a) Behandlung von fehlenden Werten*). Die Unterscheidung liegt lediglich in der Bezeichnung der Funktionen und die damit verbundenen verschiedenen Parametern, z. B. ist beim Clipping der Parameter, der den prozentualen Anteil der Daten als Ausreißer angibt, entscheidend. Wird dieser Parameter z. B. auf 5% gesetzt, dann bedeutet das, dass 10% des Wertebereichs (5% oben und 5% unten) Ausreißer sind. Beispielhaft ist die Behandlung von Ausreißern in Listing 3.4 mit Winsorizing dargestellt.

```
-- Werte für die Attribute (5% oben und 5% unten)  werden ermittelt
-- Ausgenommen sind die Attribute id und column_a
DBMS_DATA_MINING_TRANSFORM.INSERT_MISS_NUM_MEAN(
       'winsor_def_table', -- Clipping-Tabelle
       'data_table',       -- Datentabelle
       0.05,               -- prozent. Anteil der betroffenen Daten
       DBMS_DATA_MINING_TRANSFORM.Column_List{'id', 'column_a'}
);
```

Listing 3.4: Behandlung von Ausreißern

2.) One-Class Support-Vector-Machine:

In Oracle Data Mining kann speziell eine weitere Möglichkeit für die Behandlung von Ausreißern genutzt werden. Es handelt sich dabei um den One-Class Support-Vector-Machine-Algorithmus, der eine Variante des Support-Vector-Maschine-Algorithmus (siehe Kapitel 2.6.4 (Methoden zur Klassifikation)) darstellt.

Diese Methode zur Identifikation von Ausreißern wird in ODM als Anomaly-Detection (Anomalie-Erkennung) bezeichnet. Dabei wird ein sogenanntes Profil auf Basis der Daten erstellt, welches die Daten in zwei Gruppen (*normale* Daten und *andere* Daten, sogenannte Ausreißer) einteilt, und in einem Modell-Objekt, das die Unterscheidung zwischen diesen beiden Gruppen enthält, gespeichert. Anschließend wird dieses Profil für neue oder bestehende Daten verwendet, um diese einer der beiden Gruppen zuzuordnen. Eine direkte Behandlung der Ausreißer mit der Anomalie-Erkennung ist jedoch nicht möglich, da diese lediglich identifiziert werden. Die Behandlung, d. h. die Korrektur, muss entsprechend anderweitig erfolgen.

Anmerkung: Die Verwendung von Data Mining-Methoden zur Behandlung oder Identifikation von Ausreißern unterscheidet sich ein wenig von den anderen Methoden, wie z. B. dem Clipping. Das Clipping identifiziert und behandelt Ausreißer jeweils für ein einzelnes Attribut, z. B. werden die Attribute `Alter` und `Einkommen` separat voneinander betrachtet. Der One-Class Support-Vector-Machine-Algorithmus ist eine Art Klassifikation und funktioniert auf dem gesamten Datensatz, wodurch dieser als Ganzes bewertet und identifiziert wird, z. B. werden die Attribute `Alter` und `Einkommen` zusammen betrachtet und im Anschluss entschieden, ob es sich um einen Ausreißer handelt.

Um den One-Class Support-Vector-Machine-Algorithmus für die Identifikation und somit für die Behandlung von Ausreißern nutzen zu können, müssen die nachfolgenden Schritte durchgeführt werden, die zusätzlich noch in Listing 3.5 demonstriert sind:

1. Klassifikation als Data Mining-Funktion spezifizieren
2. Support Vector Machines als Klassifikationsalgorithmus spezifizieren
3. Klassifikationsattribut auf `NULL` setzen
4. Konstruktion des Modells zur Klassifizierung von Anomalien
5. Neue bzw. existierende Daten lassen sich unter Anwendung des Modells als Ausreißer respektive Anomalien identifizieren (Vorhersage: 1 für 'Normal' und 0 für 'Ausreißer')

```
BEGIN
 -- Algorithmus ist One-Class Support-Vector-Machines
 INSERT INTO ANOMALY_DETECTION_SETTINGS (setting_name, setting_value)
 VALUES (DBMS_DATA_MINING.algo_name,
         DBMS_DATA_MINING.support_vector_machine);

 -- Profil zur Identifizierung von Ausreißern erstellen
 DBMS_DATA_MINING.CREATE_MODEL(
     model_name          => 'ANOMALY_DETECTION_MODEL',
     mining_function     => DBMS_DATA_MINING.classification,
     target_column_name  => NULL,     -- One-Class benötigt 'NULL'
     data_table_name     => 'data',
     case_id_column_name => 'object_id',
     settings_table      => 'ANOMALY_DETECTION_SETTINGS');
END; /

 -- Unter Verwendung der SQL-Funktion PREDICTION_PROBABILITY
 -- und PREDICTION lassen sich Ausreißer einer Datenmenge identif.
 SELECT ID,
        PREDICTION(ANOMALY_DETECTION_MODEL using *) as Prediction,
        PREDICTION_PROBABILITY(ANOMALY_DETECTION_MODEL,
                               0  using *) as Probability
 FROM data_table
 ORDER BY 1;

        ID           PREDICTION             PROBABILITY
       -----        ----------             -----------
                          ...
       1001             1                      .93
       1001             0                      .07
       1002             1                      .67
       1002             0                      .33
                          ...
```

Listing 3.5: Modell-Objekt zur Anomalie-Erkennung

Eine weitere Möglichkeit zur Identifikation der Ausreißer neben der Verwendung der SQL-Funktion `PREDICTION_PROBABILITY` besteht in der Benutzung der Methode `DBMS_DATA_MINING.APPLY` (Listing 3.6). Diese Methode bekommt als Eingabe eine Tabelle, für die die Ausreißer identifiziert werden sollen, und liefert als Ausgabe eine Tabelle, die das Ergebnis für jeden Datensatz enthält. Das Schema der Tabelle, die das Ergebnis speichert, ist im Folgenden dargestellt:

Spaltenname	Datentyp	Bedeutung
id	VARCHAR2/NUMBER	Identifier
prediction	NUMBER	Vorhersage
probability	NUMBER	Wahrscheinlichkeit der Vorhersage

Das Attribut `prediction` kann dabei zwei Werte annehmen – 0 steht für Ausreißer und 1 steht für typischer Wert. Und das Attribut `probability` gibt die Wahrscheinlichkeit für die getroffene Vorhersage an. In Listing ref wird die Anwendung demonstriert.

```
BEGIN
  -- Identifizierung unter Verwendung der APPLY-Funktion
 DBMS_DATA_MINING.APPLY(
     model_name          => 'ANOMALY_DETECTION_MODEL',
     data_table_name     => 'data',
     case_id_column_name => 'object_id',
     result_table_name   => 'outlier_apply_result'
 );

 -- Ausgabe des Ergebnisses der Ausreißer-Analyse
 SELECT * FROM outlier_apply_result

     id              prediction            probability
    -----            ----------            -----------
                        ...
    1001                 1                     .93
    1001                 0                     .07
                        ...
END; /
```

Listing 3.6: Anomalie-Erkennung mit APPLY-Methode

Anmerkung: Eine Behandlung von Ausreißern mittels Clustering bzw. Klassifikation ist im Vergleich zum Clipping sehr aufwendig, da diese Methode wie die Data Mining-Methoden ein eigenes Preprocessing und eine eigene Transformation der Daten benötigt, und sollte möglichst nur dann eingesetzt werden, wenn explizit nach Ausreißern gesucht wird bzw. wenn sich dadurch ein wesentlich besseres Ergebnis erzielen lässt. Außerdem lassen sich damit nur ganze Datensätze als Ausreißer identifizieren, eine Behandlung von einzelnen Attributen ist nicht möglich. Im Sinne der Behandlung der Ausreißer ist daher das Clipping zu verstehen.

3.5.2 Oracle Transformation

Nach dem Preprocessing können die Daten, bevor diese dem Data Mining-Schritt übergeben werden, transformiert werden. Die Transformation ist wie bereits das Preprocessing in Oracle optional. In ODM sind die Funktionen für die Transformation im Package `DBMS_DATA_MINING_TRANSFORM` enthalten. Weitere Methoden, die ebenfalls zur Transformation der Daten verwendet werden können, sind im Package `DBMS_DATA_MINING` zu finden.

Anmerkung: Die Methoden für die Transformation im Package `DBMS_DATA_MINING` unterscheiden sich von denen im Package `DBMS_DATA_MINING_TRANSFORM` dahingehend, dass diese die gleichen Vorbereitungen, d. h. eigenes Preprocessing und Transformation (beschränkt auf die Funktionen aus dem Package `DBMS_DATA_MINING_TRANSFORM`) benötigen wie die Data Mining-Methoden.

Von den in Kapitel 2.5 vorgestellten Techniken zur Transformation der Daten sind die folgenden Methoden in ODM umgesetzt worden:

(a) Normierung (Package: `DBMS_DATA_MINING_TRANSFORM`)
(b) Diskretisierung (Package: `DBMS_DATA_MINING_TRANSFORM`)
(c) Attribut-Reduktion (Package: `DBMS_DATA_MINING`)

(a) Normierung:

Im ersten Schritt der Normierung wird eine Tabelle (Normierungstabelle) angelegt, die die Werte für die eigentliche Normierung der Attribute speichern soll (die Werte werden zur Transformation in einen neuen Wertebereich benutzt). Es werden der Name des Attributs (`col`), die Verschiebung (`shift`) um einen errechneten Wert und ein Skalierungsfaktor (`scale`) gespeichert. Bei der Rücktransformation der normierten Werte werden die Informationen der Normierungstabelle wiederum benötigt, um die ursprünglichen Werte zurückzurechnen. Diese Tabelle hat das folgende Schema:

Spaltenname	Datentyp	Bedeutung
col	VARCHAR2(30)	Name des Attributs
shift	NUMBER	Verschiebung (T. d. Zählers)
scale	NUMBER	Skalierung (Nenner)

Im zweiten Schritt der Normierung werden die einzelnen Werte für `shift` und `scale` der numerischen Attribute je nach Normierungsmethode ermittelt. ODM stellt folgende Arten der Normierung zur Verfügung (siehe Kapitel 2.5):

- **Min-Max-Normierung:** $shift = Minimum$ und $scale = Maximum - Minimum$
- **Z-Score-Normierung:** $shift = Mittelwert$ und $scale = Standardabweichung$
- **Skalen-Normierung:** $shift = 0$ und $scale = max\{abs(Maximum), abs(Minimum)\}$

Im dritten Schritt wird die Normierung der numerischen Attribute abgeschlossen, indem eine Sicht auf der Datentabelle, unter Verwendung der Informationen in der Normierungstabelle, erzeugt wird. Für die Normierung wird die folgende Berechnungsvorschrift benutzt: $norm\_x = \frac{(x-shift)}{scale}$. Für die Min-Max-Normierung ergibt sich die folgende Berechnung: $norm\_x = \frac{(x-Minimum)}{(Maximum-Minimum)}$.

In Listing 3.7 wird die Durchführung der Normierung von numerischen Werten in ODM beschrieben. In diesem Beispiel wird die Min-Max-Normierung (Methode `INSERT_NORM_LIN_MINMAX`) zur Normierung der numerischen Wert benutzt.

```
BEGIN
 -- Definition der Tabelle zur Normierung von Werten
 -- (Tabelle speichert die Werte (shift und scale) der Attribute,
 -- die im zweiten Schritt ermittelt werden)
 DBMS_DATA_MINING_TRANSFORM.CREATE_NORM_LIN('norm_def_table');

 -- Werte für die Attribute (d.h. shift und scale) werden ermittelt
 -- (Min-Max-Normierung)
 -- Ausgenommen von der Normierung sind die Attribute id und column_a
 DBMS_DATA_MINING_TRANSFORM.INSERT_NORM_LIN_MINMAX(
        'norm_def_table',   -- Normierungstabelle
        'data_table',       -- Datentabelle
        DBMS_DATA_MINING_TRANSFORM.Column_List{'id', 'column_a'}
 );
 -- Erstellt eine Sicht auf Basis der Datentabelle, in der die
 -- Werte normiert wurden
```

```
 DBMS_DATA_MINING_TRANSFORM.XFORM_NORM_LIN(
        'norm_def_table',   -- Werte für Shift und Scale
        'data_table',       -- Datentabelle
        'norm_view');       -- Ergebnis: Sicht
END; /
```

Listing 3.7: Normierung numerischer Werte

(b) Diskretisierung:

Im ersten Schritt der Diskretisierung wird eine Tabelle (Diskretisierungstabelle) angelegt, die die Werte für die eigentliche Diskretisierung der Attribute speichern soll. Die Diskretisierungstabelle wird wiederum zur Rücktransformation der diskretisierten Werte (wenn auch nicht verlustfrei) benutzt. Diese Tabelle hat das folgende Schema:

Spaltenname	Datentyp	Bedeutung
col	VARCHAR2(30)	Name des Attributs
val	VARCHAR2(4000) bzw. NUMBER	nominaler Wert bzw. untere Grenze
bin	VARCHAR2(4000)	diskretisierter Wert (Behälter)

Im zweiten Schritt der Diskretisierung werden die einzelnen Werte für `bin` (Behälter) und `val` (unterer Grenzwert des Behälters) der Attribute ermittelt. ODM stellt folgende Arten der Diskretisierung (siehe Kapitel 2.5) zur Verfügung:

- **Top-N frequency Binning** für nominale Werte
- **(automatisches) Equi-width Binning** für numerische Werte (wahlweise mit automatischer Berechnung der optimalen Menge diskreter Werte auf Basis von Statistiken der Datenmenge)
- **Equi-height Binning** für numerische Werte (in Oracle **Quantile Binning** genannt)

Im dritten Schritt wird die Diskretisierung der Attribute abgeschlossen, indem eine Sicht auf der Datentabelle, unter Verwendung der Informationen in der Diskretisierungstabelle, erzeugt wird.

In Listing 3.8 ist die Diskretisierung numerischer Werte verkürzt dargestellt. Bei der Diskretisierung ist der Parameter `bin_num`, der die Anzahl der diskreten Werte angibt, in der Funktion `INSERT_BIN_NUM_EQUIWIDTH` für Equi-width Binning, wichtig.

```
-- Werte für die Attribute (d.h. bin und val) werden ermittelt
-- Anzahl der diskreten Werte wird mit 'bin_num' angegeben
-- Ausgenommen sind die Attribute id und column_a
DBMS_DATA_MINING_TRANSFORM.INSERT_BIN_NUM_EQUIWIDTH(
        'bin_def_table',    -- Diskretisierungstabelle
        'data_table',       -- Datentabelle
        'bin_num',          -- Anzahl diskreter Werte
        DBMS_DATA_MINING_TRANSFORM.Column_List{'id', 'column_a'} );
```

Listing 3.8: Diskretisierung numerischer Werte

(c) Attribut-Reduktion:

ODM stellt für die Attribut-Reduktion zwei Methoden – `Attribute Importance` (Attribut-Wichtigkeit) und `Feature-Extraction` (Extraktion von Eigenschaften) zur Verfügung.

Anmerkung: Da die Methoden `Attribute Importance` und `Feature-Extraction` nach dem Prinzip des Modell-Objekts funktionieren, kann ein vorheriges Preprocessing und eine Transformation vorgenommen werden. Das Ergebnis der Attribut-Reduktion kann anschließend dazu genutzt werden, um die Daten um irrelevante und nur schwach-relevante Attribute zu minimieren.

Attribute Importance: Die `Attribute Importance`-Methode, auch `Feature-Selection` genannt, ermittelt die Relevanz von Attributen in Bezug auf ein anderes Attribut, z. B. kann das Attribut `Beruf` gewissermaßen das Attribut `Einkommen` beschreiben. Wirkt sich ein Attribut nicht entscheidend genug auf das vorgegebene Attribut aus, also trägt es nichts zu dessen Beschreibung bei, dann kann es auch weggelassen werden, ohne dass ein Nachteil entsteht. ODM verwendet für die Bestimmung der `Attribute Importance` die Minimum Description Length[Ris78].

In Listing 3.9 wird die `Attribute Importance` für die Bewertung eines Attributs mit der Bezeichnung *'target'* gemessen. Dabei wird ein Modell-Objekt erstellt, in dem die Informationen gespeichert werden. Es sei angenommen, dass ein Preprocessing und eine Transformation der Daten bereits durchgeführt worden ist. Durch die Vorverarbeitung der Daten, speziell durch die Diskretisierung, kann das Ergebnis der `Attribut Importance` verbessert werden, da dadurch der Einfluss der einzelnen Attribute in Bezug auf das Attribut mit der Bezeichnung *'target'* deutlich hervorgehoben wird. Ansonsten sind für den `Attribute Importance`-Algorithmus keine Einstellungen vorzunehmen.

```
-- Berechnung der Attribute-Importance für das Attribut 'target'
DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'MODEL_AI',
    mining_function     => DBMS_DATA_MINING.attribute_importance,
    data_table_name     => 'data',
    case_id_column_name=> 'object_id',
    target_column_name  => 'target' );
```

Listing 3.9: Berechnung der Attribute Importance

Das Ergebnis der `Attribute Importance` kann über die Funktion `GET_MODEL_DETAILS_AI` abgerufen werden. Diese Funktion liefert eine Auflistung der Attribute und deren Einfluss auf das zuvor angegebene Attribut. Dabei drücken positive Werte eine direkten Einfluss aus und negative Werte keinen Einfluss aus. Eine Auflistung ist beispielhaft in Listing 3.10 dargestellt.

```
-- Berechnung der Attribute-Importance für das Attribut 'target'
SELECT attribute_name, importance_value, rank
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_AI('model_ai'))
ORDER BY rank;

ATTRIBUTE_NAME              IMPORTANCE_VALUE        RANK
--------------------------- ----------------    --------
Attribut_A                     0.537029338           1
Attribut_B                     0.443330186           2
Attribut_C                     0.371838964           3
Attribut_D                    -0.115636359           4
Attribut_E                    -0.354888343           5
```

Listing 3.10: Ergebnis der Attribute Importance

Feature-Extraction: Die `Feature-Extraction`-Funktion erstellt auf Grundlage der Basisdaten eine Menge von Faktoren, sogenannte Eigenschaften, die die wichtigsten Informationen beinhalten, um die Basisdaten zu charakterisieren. Die Faktoren können also mehrere Attribute in sich vereinigen (z. B. dadurch dass ein Faktor einer Linearkombination dem ursprünglichen Attri-

bute entspricht) und dadurch die notwendige Menge der Attribute zur Beschreibung der Basisdaten reduzieren. Die Faktoren, die die Daten beschreiben, sind auf einen geringen Bruchteil reduzierbar, ohne das eine signifikante Ungenauigkeit entsteht, wodurch Attribute (mit geringem Anteil am Faktor) weggelassen werden können. ODM implementiert die `Feature-Extraction` mittels des Non-Negative Matrix Factorization [DL99].

In Listing 3.11 wird eine Feature-Extraction durchgeführt. Es sei angenommen, dass ein Preprocessing und eine Transformation der Daten bereits durchgeführt worden ist.

```
-- Berechnung der Feature Extraction für das Attribut 'target'
DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'MODEL_FE',
    mining_function     => DBMS_DATA_MINING.feature_extraction,
    data_table_name     => 'data',
    case_id_column_name => 'object_id');
```

Listing 3.11: Berechnung der Feature-Extraction

Das Ergebnis der `Feature-Extraction` ist eine Menge von Faktoren, die sich aus mehreren Attributen der Basisdaten zusammensetzen und sich über die Funktion `GET_MODEL_DETAILS_NMF` abrufen lassen. Den Attributen in den Faktoren sind Koeffizienten zugeordnet, die ausdrücken, welchen Anteil sie am Faktor haben.
In ODM werden numerische und nominale Attribute unterschiedlich behandelt. Die nominalen Attribute werden als Name-Wert-Paare beschrieben, d. h. jeder Wert eines nominalen Attributes erhält seinen eigenen Koeffizienten, und die numerischen Attribute werden nur mit dem Namen beschrieben. Eine Auflistung der Faktoren ist beispielhaft in Listing 3.12 dargestellt, wobei es sich ausschließlich um numerische Attribute in den Faktoren handelt.

```
-- Berechnung der Feature-Extraction für das Attribut 'target'
SELECT feature_id, attribute_name, attribute_value, coefficient
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_NMF('model_nmf'))F,
TABLE(F.attribute_set) A
ORDER BY feature_id, coefficient;

 FEATURE_ID  ATTRIBUTE_NAME    ATTRIBUTE_VALUE    COEFFICIENT
 ----------  --------------    ---------------    -----------
 1           Attribut_A                            5.6996389
             Attribut_B                            0.9815716
             Attribut_C                           86.7749359
             Attribut_D                            1.6792973
 2           Attribut_A                            7.4687617
             Attribut_B                           20.929389
 3           Attribut_A                           43.3020941
             Attribut_B                            5.6996389
 4           Attribut_E          Wert_1           12.3020941
             Attribut_F          Wert_2            5.6996389
```

Listing 3.12: Ergebnis der Feature-Extraction

Durch eine Analyse der Faktoren in Hinblick auf die Attribute und deren Anteil am Faktor, lassen sich unwichtige Attribute der Basisdaten identifizieren und somit eine Attribut-Reduktion durchführen.

In ODM werden zusätzlich die SQL-Funktionen `FEATURE_ID`, `FEATURE_SET` und `FEATURE_VALUE` [Ora06h] zur Verfügung gestellt, um die Faktoren zur Beschreibung von unbekannte Daten zu verwenden. Die Funktion `FEATURE_ID` ermittelt den Faktor und `FEATURE_VALUE` ermittelt den exakten Wert der Übereinstimmung mit dem Faktor, der den unbekannten Datensatz am besten beschreibt. Und schließlich ermittelt die Funktion `FEATURE_SET` eine Menge von Faktor-Wert-Paaren (basierend auf `FEATURE_ID` und `FEATURE_VALUE`) für einen unbekannten Datensatz. Für weitere und detailliertere Informationen zur Benutzung der Funktionen wird auf die Dokumentation von Oracle verwiesen [Ora06h].

3.5.3 Oracle Data Mining

ODM bietet eine breite Auswahl an Techniken an, um Data Mining für unterschiedliche Aufgaben und Interessen zu betreiben. Im Vordergrund stehen die Methoden und Techniken, die in Kapitel 2 beschrieben wurden. Es folgt eine Auflistung der Data Mining-Methoden und der konkreten Verfahren, die in der Data Mining-Cartridge integriert sind:

- Clustering (deskriptiv)
  - Enhanced k-Means Clustering
  - Orthogonal Partitioning Clustering
- Assoziationsanalyse (deskriptiv)
  - Apriori-Algorithmus
- Klassifikation (prädiktiv)
  - Klassifikationsbaum
  - Naive Bayes Klassifikation
  - Adaptives Bayessches Netzwerk
  - Support Vector Machines (One-class und Multi-class)
- Regression (prädiktiv)
  - Support Vector Machines

Clustering

In ODM sind für das Clustering zwei Verfahren implementiert. Zum einen den Enhanced k-Means Clustering-Algorithmus in Anlehnung an den bekannten k-Means-Algorithmus, der jedoch einen hybriden Ansatz zwischen partitioning-based (siehe Partitioning-based Methods in Kapitel 2.6.2) und hierarchical-based Methode (siehe Partitioning-based Methods in Kapitel 2.6.2) darstellt. Zum anderen eine proprietäre Entwicklung von Oracle mit der Bezeichnung Orthogonal Partitioning Clustering Algorithmus (kurz: O-Cluster), der einen hybriden Ansatz zwischen grid-based (siehe Grid-based Methods in Kapitel 2.6.2), density-based (siehe Density-based Methods in Kapitel 2.6.2) und hierarchical-based (siehe Partitioning-based Methods in Kapitel 2.6.2) Methode darstellt. In Abb. 3.3 ist der schematische Ablauf des Clusterings in Oracle illustriert.

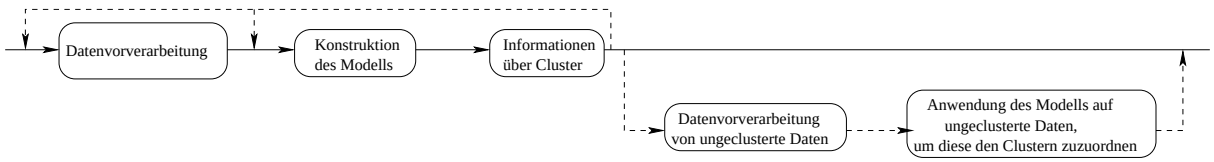

Datenvorverarbeitung
Konstruktion des Modells
Informationen über Cluster
Datenvorverarbeitung von ungeclusterte Daten
Anwendung des Modells auf ungeclusterte Daten, um diese den Clustern zuzuordnen

Abbildung 3.3: Flussdiagramm: Clustering

Enhanced k-Means Clustering:

Der Enhanced k-Means-Algorithmus geht bei der Konstruktion der Cluster hierarchisch nach dem divisiven Prinzip vor. Es wird dabei eine binäre Cluster-Hierarchie (d. h. binärer Baum) aufgebaut, bei der in jedem Schritt nur jeweils ein Cluster der Hierarchie in zwei neue Cluster aufgeteilt wird. Diese Aufteilung der Cluster bzw. Konstruktion der Cluster-Hierarchie wird solange durchgeführt, bis die angegebene Anzahl an Clustern erreicht ist.

Das Aufteilungskriterium bestimmt dabei, welcher Cluster der Hierarchie im nächsten Schritt in zwei neue Cluster aufgeteilt werden soll, so dass die Cluster möglichst homogen sind. Bei der Aufteilung eines Clusters kommt der partitionierende Charakter des Algorithmus zum Tragen, in dem die Daten auf Ähnlichkeit bewertet werden.

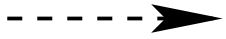
Für die Ähnlichkeit von zwei Daten(sätzen) (Objekten) wird eine Abstandsfunktion im k-Means Algorithmus verwendet. ODM stellt dafür drei Abstandsfunktionen (1. Euklidischer-Abstand, 2. Cosinus-Abstand und 3. Schneller-Cosinus-Abstand (siehe Kapitel 2.6.2)) bereit, die alle lediglich numerische Attribute zur Berechnung akzeptieren. Da das Clustering mittels Enhanced k-Means jedoch auch für nicht-numerische Attribute funktionieren soll, werden die nicht-numerischen Attribute intern durch eine Menge von binären Attributen $A_i \in \{0, 1\}$ ersetzt. (Diese Behandlung von nicht-numerischen Attributen erklärt auch die fehlende Unterstützung von Ähnlichkeitsmatrizen (siehe Kapitel 2.6.2) vom Algorithmus.)

Beispiel: Es sei das nicht-numerische Attribut `Geschlecht G` mit `G` $\in \{m, w\}$ gegeben. Intern wird `Geschlecht` auf die neuen Attribute `G_m` für Geschlecht=männlich und `G_w` für Geschlecht=weiblich abgebildet. In Abb. 3.4 ist die interne Transformation anschaulich dargestellt und folgende Berechnung zur Bewertung der Ähnlichkeit wurde durchgeführt:

Der Euklidische-Abstand zwischen den Datensätzen `id=5` und `id=7` berechnet sich zu $\approx 7,14$ und zwischen den Datensätzen `id=6` und `id=7` zu ≈ 12, woraus folgt, dass `id=5` und `id=7` zueinander ähnlicher sind als `id=6` und `id=7`.

Es ist zu beobachten, dass trotz der Transformation nicht-numerischer Attribute und die damit verbundene Einbeziehung in die Ähnlichkeitsberechnung die Ähnlichkeit von den numerischen Attributen dominiert wird. Aus diesem Grund müssen numerische Attribute egal, ob diese in Kombination mit nicht-numerischen Attributen betrachtet werden, stets im Transformationsschritt normiert werden, um eben eine Dominanz von numerischen Attributen zu verhindern. Beispielsweise würde die Normierung des `Alters` auf den Bereich $[0, 1]$ bewirken, dass nun die Datensätze `id=6` und `id=7` ähnlicher sind als die Datensätze `id=5` und `id=7`.

ID	Alter	Geschlecht G
...	...	...
5	23	m
6	42	w
7	30	w
...	...	...



ID	Alter	G_m	G_w
...	...	...	...
5	23	1	0
6	42	0	1
7	30	0	1
...	...	...	...

Abbildung 3.4: interne Umwandlung von Zeichenketten

Die Parameter und Einstellungen, die für den Enhanced k-Means Algorithmus vorgenommen werden können, werden im Folgenden beschrieben und in der nachfolgenden Tabelle zusammenfassend aufgelistet.

Die Parameter `kmns_iterations` und `kmns_conv_tolerance` haben einen Einfluss auf die Abbruchbedingung und somit auf die Qualität und die Laufzeit des Algorithmus. Der Parameter `kmns_distance` bestimmt nach welcher Methode die Ähnlichkeit zwischen Daten gemessen werden soll.

Und der Parameter `kmns_split_criterion` gibt das Kriterium an, welcher Cluster als nächstes aufgeteilt wird. Für die meisten Clustering-Aufgaben liefern die Voreinstellungen (Standardwerte) recht gute Ergebnisse. Im Allgemeinen ist lediglich der Parameter `clus_num_clusters` zu spezifizieren, da dieser die Anzahl der zu erzeugenden Cluster angibt.

Die Parameter `kmns_num_bins` und `kmns_min_pct_attr_support` haben keinen Einfluss auf die Konstruktion der Cluster. Diese sind bei der Analyse und zur Beschreibung der Cluster von Bedeutung. Nach dem Konstruktionsprozess steht für jeden Cluster und jedes Attribut eine Statistik in Form von einem Histogramm zur Verfügung. Der Parameter `kmns_num_bins` spezifiziert dabei die max. Anzahl an diskreten Werten im Histogramm. Dieser Wert gilt ausschließlich für numerische Attribute, bei nicht-numerischen Attributen ist die Anzahl an diskreten Werten durch die verschiedenen Werte des Attributs bestimmt. Außerdem wird jeder Cluster durch eine Regel der Form `IF-THEN` beschrieben. Und der Parameter `kmns_min_pct_attr_support` gibt dabei an, wann ein Attribut in eine Beschreibungsregel aufgenommen werden soll (d. h. Relevanz des Attributes zur Beschreibung des Clusters).

Für die Verwendung des Enhanced k-Means-Algorithmus gibt Oracle noch folgende Empfehlungen, um möglichst gute Resultate zu erzielen [Ora06d]:

- numerische Werte normalisieren
- fehlende Werte mit Mittelwerten auffüllen
- numerische und nicht-numerische Attribute diskretisieren
- Behandlung von Ausreißer

Die Parameter und Einstellungen für den Enhanced k-Means-Algorithmus sind der folgenden Tabelle zu entnehmen, wobei die Standardwerte **fett** hervorgehoben sind.

Parameter	Werte	Beschreibung
`algo_name`	`algo_kmeans`	Spezifiziert den k-Means als Clustering Algorithmus, den das Modell verwenden soll
`clus_num_clusters`	$x \geq 1$, $x = 10$	Anzahl an Cluster, die erstellt werden sollen
`kmns_distance`	**`kmns_euclidean`**, `kmns_cosine`, `kmns_fast_cosine`	Distanz- bzw. Abstandsfunktion für den k-Means Clustering Algorithmus
`kmns_iterations`	$x \in [0, 20]$, $x = 3$	Anzahl der durchzuführenden Iterationen
`kmns_conv_tolerance`	$x \in [0, 0.5]$, $x = 0.01$	Konvergenz-Toleranz
`kmns_split_criterion`	**`kmns_variance`**, `kmns_size`	Aufteilungskriterium
`kmns_num_bins`	$x \in [0, \infty]$, $x = 10$	Anzahl der Behälter (Bin) im Histogramm. Die Grenzen der Bins werden global auf der Trainingsmenge berechnet.
`kmns_block_growth`	$x \in [1, 5]$, $x = 2$	Wachstumsfaktor für den allozierten Speicher, der einen Cluster enthält
`kmns_min_pct_attr_support`	$x \in [0, 1]$, $x = 0.1$	Mindestprozentsatz eines Attributs, um in der Beschreibungsregel des Clusters zu stehen

Orthogonal Partitioning Clustering:

Der O-Cluster-Algorithmus geht bei der Konstruktion der Cluster ebenfalls hierarchisch nach dem divisiven Prinzip vor, wobei wie beim Enhanced k-Means-Algorithmus eine binäre Cluster-Hierarchie (d. h. binärer Baum) aufgebaut wird. Diese Aufteilung der Cluster respektive das Wachstum des Baumes wird solange durchgeführt, bis entweder alle Daten untersucht worden sind oder keine signifikanten Verbesserungen durch neue Cluster zu erwarten sind. Dabei arbeitet der Algorithmus stets auf einer kleinen Menge von Daten, deren Größe durch den Parameter *max_buffer* spezifiziert ist.

Die Cluster, die durch den O-Cluster-Algorithmus entstehen, entsprechen einer Gitterstruktur, ähnlich der bei den grid-based Methoden. Die Konstruktion eines neuen Clusters wird dabei durch ein Aufteilungskriterium, das nach der density-based Methode funktioniert, entschieden. Dabei wird versucht in einem Cluster einen Bereich, auch als Tal bezeichnet, mit der niedrigsten Dichte umgeben von Bereichen mit höheren Dichten, zu finden, so dass der Unterschied maximal ist. An dieser Stelle wird der Cluster dann aufgeteilt. Für die beiden neuen Cluster wird anschließend ebenfalls eine Aufteilung gesucht (rekursiver Aufruf des Algorithmus mit den neu entstandenen Clustern) bis keine sinnvollen Aufteilungen mehr möglich sind. Für eine detailliertere Beschreibung des Algorithmus sei auf [MC02] verwiesen.

Der O-Cluster benötigt im Vergleich zum Enhanced k-Means-Algorithmus eigentlich keine Parameter, jedoch hat sich herausgestellt, dass die Einführung eines Parameters *sensitivity* nützlich sein kann [MC02]. Der Parameter *sensitivity* hat die Aufgabe die Anzahl der Cluster zu regulieren. Ein hoher Wert für *sensitivity* hat dabei zur Folge, dass viele Cluster, darunter viele kleine Cluster, konstruiert werden. Bei einem kleinen Wert für *sensitivity* werden hingegen insgesamt weniger Cluster konstruiert, da die kleinen Cluster herausgefiltert werden.

Bei der Verwendung des O-Cluster-Algorithmus gibt Oracle noch folgende Empfehlungen, um möglichst gute Resultate zu erzielen:

- Behandlung der Ausreißer durch Trimming, nicht durch Winsorizing
- Diskretisierung numerischer Werte, da O-Cluster kein distanz-basierter Algorithmus ist, und ansonsten keine optimalen Aufteilungen möglich sind

Die Einstellungen, die für den O-Cluster-Algorithmus vorgenommen werden können, sind in der folgenden Tabelle aufgelistet.

Parameter	Werte	Beschreibung
`algo_name`	`algo_o_cluster`	Spezifiziert den Clustering Algorithmus
`oclt_max_buffer`	$x \in [0, \infty], x = 50000$	Puffergröße
`oclt_sensitivity`	$x \in [0, 1], x = 0.5$	Anteil der angibt, wann ein neuer Cluster erstellt wird.

Clustering mit Oracle Data Mining:

Es soll im Folgenden der Prozess des Clustering (siehe Abb. 3.3) exemplarisch demonstriert werden. Dazu ist in Listing 3.13 ein Code-Ausschnitt gegeben, der das Clustering mit dem k-Means-Cluster-Algorithmus illustriert, mit dem Ziel, dass im Anschluss drei Cluster entstehen. In Abb. 3.5 ist die korrespondierende Punktmenge (schwarze Punkte) abgebildet, die in drei Cluster unterteilt werden soll. Die Einstellungen für den Clustering-Algorithmus sind in der Konfigurationstabelle `KM_SETTINGS` gespeichert.

```
CREATE TABLE KM_SETTINGS(
  setting_name    VARCHAR2(30),
  setting_value   VARCHAR2(128) );
BEGIN
  -- Einstellungen: k-Means-Algorithmus und Anzahl der Cluster ist 3
  INSERT INTO KM_SETTINGS (setting_name, setting_value)
      VALUES (DBMS_DATA_MINING.algo_name,
              DBMS_DATA_MINING.algo_kmeans);
  INSERT INTO KM_SETTINGS (setting_name, setting_value)
      VALUES (DBMS_DATA_MINING.clus_num_clusters, 3);
END; /
-- Data Mining mit der Clustering-Methode
DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'KM_MODEL',
    mining_function     => DBMS_DATA_MINING.clustering,
    data_table_name     => 'data',
    case_id_column_name => 'object_id',
    setting_table_name  => 'KM_SETTINGS'
);
```

Listing 3.13: Clustering mit k-Means

Im Anschluss des Clustering ist das Ergebnis im Modell-Objekt mit dem Namen `KM_MODEL` gespeichert. Das Modell-Objekt enthält, wie bereits beschrieben, eine Vielzahl an Informationen, die Aufschluss über das Ergebnis des Clustering und die entstandenen Cluster geben.

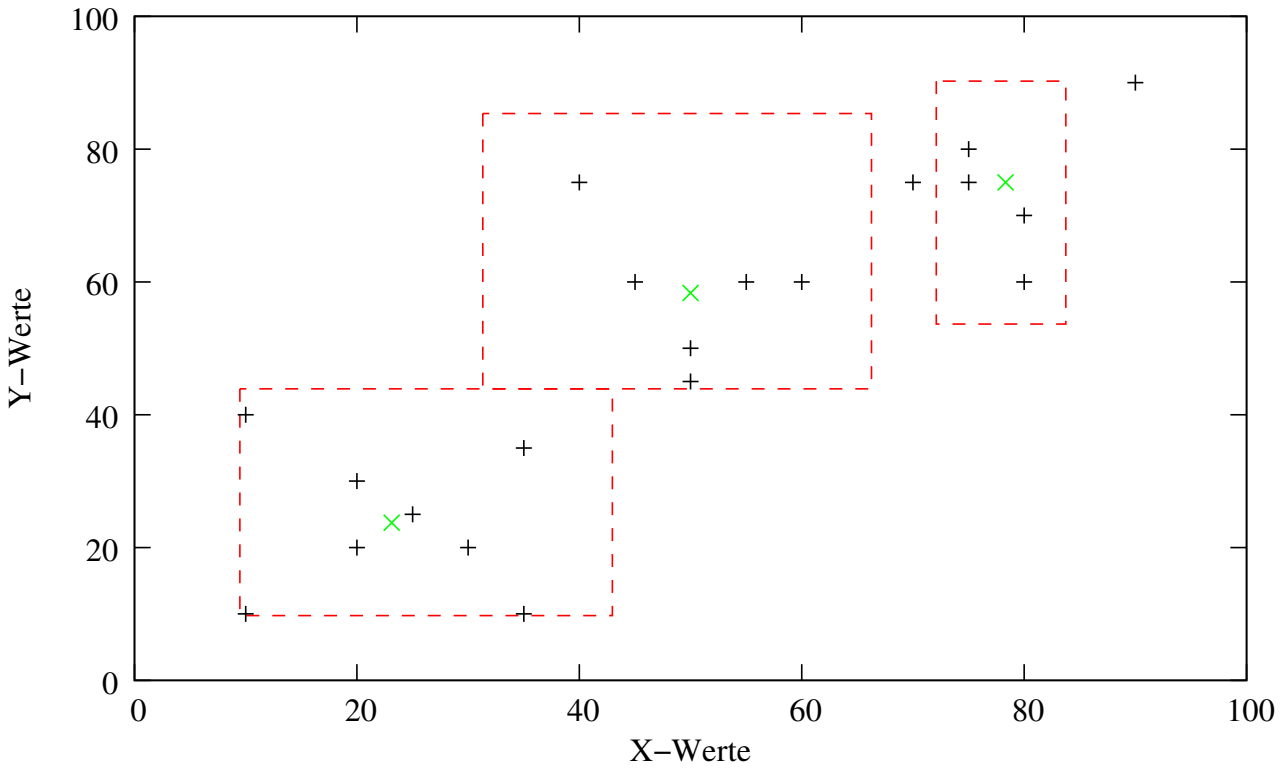


Abbildung 3.5: Beispiel: Punktmenge für das Clustering

Informationen im Clustering-Modell-Objekt:

Die Informationen, die beim Clustering gesammelt worden sind, können anschließend über die Methode `GET_MODEL_DETAILS` eingesehen werden. Für jeden Clustering-Algorithmus wird dabei eine eigene Methode bereitgestellt:

- `GET_MODEL_DETAILS_KM` für den Enhanced k-Means-Algorithmus
- `GET_MODEL_DETAILS_OC` für den O-Cluster-Algorithmus

Anmerkung: Obwohl in ODM eine Unterscheidung zwischen den `GET_MODEL_DETAILS`-Methoden gemacht wird, sind die Information, die zum einen im Modell-Objekt gespeichert sind und zum anderen über die Methoden abgerufen werden können, identisch [Ora06f]. Daher wäre es logisch diese Methoden zusammenzufassen. Die Zusammenfassung würde aber gegen das Konzept der Data Mining-Cartridge und der Strukturierung der Funktionen sprechen.

In Listing 3.14 wird die Funktion `GET_MODEL_DETAILS_KM` benutzt, um die Cluster-Hierarchie des Modells anzuzeigen. In dieser Cluster-Hierarchie sind die Cluster mit $id = 3$, $id = 4$, $id = 5$ die gesuchten Cluster, da sie die Blattknoten der Cluster-Hierarchie bilden.

```
SELECT id, parent, tree_level, record_count
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM('model_namae'));

id      parent         tree_level   record_count
-----   -----------    -----------  ------------
1                      1            20
2       1              2            12
3       1              2            8
4       2              3            6
5       2              3            6
```

Listing 3.14: Cluster-Hierarchie

In ODM werden Cluster durch die folgenden Punkte beschrieben, die ebenfalls über die oben genannten Funktionen abgerufen werden können:

- Centroid
- Statistiken in Form von Histogrammen für die Attribute
- Position in der Cluster-Hierarchie (Level in der Hierarchie und Vorgängerknoten)

Außerdem wird für jeden Cluster eine Cluster-Regel (eine Art von Klassifikationsregel) erstellt, die diesen beschreibt, z. B. `IF age > 25 AND sex = M THEN Cluster=5`. In Listing 3.15 ist eine SQL-Anfrage dargestellt, die die Centroide für die Cluster ermittelt. In Abb. 3.5 sind diese Centroide mit grünen Punkten gekennzeichnet.

```
-- Objekte im zwei-dimensionalen Raum (drei Cluster)
-- Formatierung der Ergebnisse für eine übersichtliche Darstellung
SELECT a.id, b.attribute_name || '=' || b.mean as centroid
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM('model_name'))a,
     TABLE(a.centroid) b, TABLE(a.child) c
WHERE c.id IS NULL;

id      centroid
-----   --------------------
```

```
3       (X=23.13), (Y=23.75)
4       (X=78.33), (Y=75.00)
5       (X=50.00), (Y=58.33)
```

Listing 3.15: Centroide der Cluster

In Listing 3.16 ist eine SQL-Anfrage dargestellt, die die Cluster-Regel für jeden Cluster ermittelt. In eckigen Klammern am Ende der Regel sind der Support s (=Anzahl der Daten für die diese Regel gilt) und die Confidence c (=Prozentsatz, der die Korrektheit der Regel beschreibt) der Regel zusätzlich angegeben. In Abb. 3.5 sind die Cluster-Regeln durch rote Rechtecke dargestellt.

```
-- Objekte im zwei-dimensionalen Raum (drei Cluster)
-- Formatierung der Ergebnisse für eine übersichtliche Darstellung
SELECT b.*, c.*, a.rule.rule_confidence, a.rule.rule_support
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM('model_name'))a,
     TABLE(a.rule.antecedent) b, TABLE(a.rule.consequent) c,
     TABLE(a.child) d
WHERE d.id IS NULL;

cluster-rule
--------------------
((X>=10) && (X<=42) && (Y>=10) && (Y<=42))==>(Cluster=3)[s=8, c=1]
((X>=74) && (X<=82) && (Y>=58) && (Y<=90))==>(Cluster=4)[s=4, c=.67]
((X>=34) && (X<=66) && (Y>=42) && (Y<=82))==>(Cluster=5)[s=6, c=1]
```

Listing 3.16: Cluster-Regel

Außerdem werden in Oracle die SQL-Funktionen CLUSTER_ID, CLUSTER_PROBABILITY und CLUSTER_SET [Ora06h] bereitgestellt, mit denen ungeclusterte Daten einem bestehenden Cluster-Modell zugeordnet werden können (Voraussetzung ist gleiche Vorverarbeitung, d. h. Preprocessing und Transformation). Die Funktion CLUSTER_ID bestimmt anhand von Attributen, zu welchem Cluster ein Objekt am wahrscheinlichsten gehört. Die Funktion CLUSTER_PROBABILITY berechnet die Wahrscheinlichkeit des Objekts in Bezug auf die Zugehörigkeit zu einem Cluster. Während die Funktion CLUSTER_SET eine Menge von Clustern liefert, zu denen ein Objekt potentiell gehören könnte.

Für eine ausführliche Beschreibung der Funktionen CLUSTER_ID, CLUSTER_PROBABILITY und CLUSTER_SET sei auf die Literatur [Ora06h] verwiesen.

Assoziationsanalyse

ODM implementiert für die Assoziationsanalyse nur einen Algorithmus, den Apriori-Algorithmus, um nach Assoziationsregeln zu suchen [AMS+96]. Der schematische Ablauf der Assoziationsanalyse in Oracle ist in Abb. 3.6 illustriert.

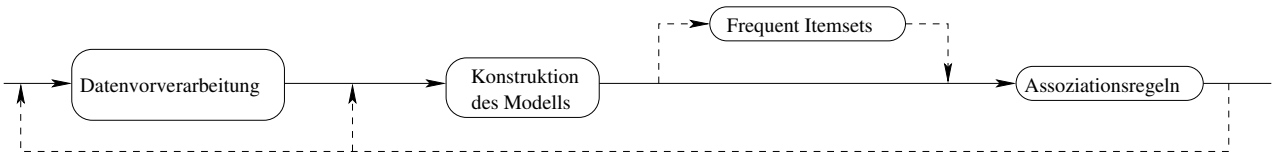


Abbildung 3.6: Flussdiagramm: Assoziationsanalyse

Es ist eine Tabelle R mit den Attributen $A1, A2, A3 \ldots$ gegeben und die Werte diese Attribute sind natürliche Zahlen. Die Assoziationsregeln in ODM haben dann den Aufbau $Ai = m \wedge \ldots \wedge Aj = n \Rightarrow Ak = o$ mit i, j, k, m, n, o beliebig gewählt. Die Elemente der Prämisse und der Konklusion der Assoziationsregeln stellen demnach Attribut-Wert-Paare dar.

Die Assoziationsanalyse in ODM ist auf Assoziationsregeln beschränkt, die die Form $A \Rightarrow B$ haben, wobei A, B Mengen darstellen und die Kardinalitäten $0 < A < 20$ und $B = 1$ besitzen. Eine Regel, dessen Konklusion aus mehreren Elementen besteht, ist folglich nicht möglich, obwohl die verwendeten Datentypen und die *frequent itemsets* in ODM vorhanden sind (siehe Datentyp `DM_RULE` für Assoziationsregel [Ora06f]). Allerdings wäre jedoch unter Verwendung der *frequent itemsets* und deren gespeicherter Support, eine Berechnung von Assoziationsregeln mit Konklusionen $B > 1$ denkbar. Die Berechnung basiert dabei auf der Definition der Confidence mit $c = \frac{s(A \Rightarrow B)}{s(A)}$ (siehe Kapitel 2.6.3).

Eine weitere Einschränkung besteht in der fehlenden Unterstützung von Konzept-Hierarchien, wodurch keine single-level und multi-level Assoziationsregeln gefunden werden können (siehe Kapitel 2.6.3).

Mit der Assoziationsanalyse in ODM können folgende Assoziationsregeln, die in Kapitel 2.6.3 vorgestellt worden sind, gefunden werden:

- eindimensionale Assoziationsregeln unter Verwendung von geschachtelten Tabellen (die Datentypen `DM_NESTED_NUMERICALS` für numerische Werte und `DM_NESTED_CATEGORICAL` für nominale Werte)
- mehrdimensionale Assoziationsregeln (normale Anwendung der Data Mining-Methode Assoziationsanalyse auf eine relationale Tabelle dessen Attribute)
- single-level und multi-level Assoziationsregeln, aber nur wenn die Konzept-Hierarchie mit immensem Aufwand durch Update-Operationen in die Daten eingepflegt werden

Die Assoziationsanalyse von eindimensionalen Assoziationsregeln ist besonders interessant, da hierbei die Verwendung von geschachtelten Tabellen notwendig ist. Bei der eindimensionalen Assoziationsanalyse geht es darum einem Datensatz mehrere Werte zuzuweisen, so dass diese als Ganzes von der Data Mining-Methode betrachtet werden. Um den Voraussetzungen von ODM zu genügen, muss also die eine mengenwertige Darstellung in eine andere mengenwertige Darstellung überführt werden, wodurch die Datensätze als Ganzes eindeutig identifizierbar werden und nach eindimensionalen Assoziationsregeln, d. h. nach Assoziationen innerhalb der Werte eines Attributes, gesucht werden kann.

In Abb. 3.7 wird als Beispiel die Warenkorbanalyse dargestellt, bei der ein Kunde mehrere Waren bei einem Einkauf erwirbt.

Die linke Darstellung entspricht der Repräsentation der *n-m*-Beziehung, wodurch jedoch keine eindeutige Identifikation der Datensätze durch ein Attribut gegeben ist, d. h. der Kunde und die von ihm gekauften Waren bilden keine geschlossene Einheit. Die rechte Darstellung verwendet eine geschachtelte Tabelle, wodurch der Kunde zur Identifikation aller seiner gekauften Waren wird, also der Einkauf durch den Kunden als Transaktion[2] dargestellt wird. Der Algorithmus in ODM verarbeitet dabei die Werte des Attributs bei der Verwendung von geschachtelten Tabellen als eigenständige Attribute. Bei der Transformation der Tabelle werden die Werte des Attributs als Attribut-Wert-Paare gespeichert, so dass wiederum die obige Darstellung der Assoziationsregel möglich ist. Bei der Warenkorbanalyse sind die Werte jedoch irrelevant und werden mit '1' für vorhanden versehen.

[2] Eine Transaktion ist eine logische Verarbeitungseinheit auf der Datenbank, die als atomare Einheit bei der Bearbeitung betrachtet wird. [EN00]

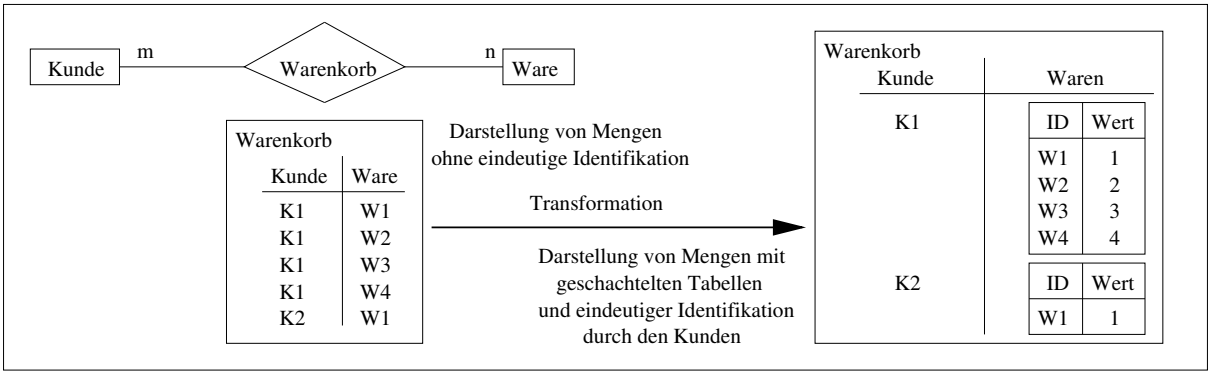


Abbildung 3.7: Mengenwertige Darstellung durch geschachtelte Tabellen

Die Parameter und Einstellungen, die für den Apriori-Algorithmus vorgenommen werden können, werden im Folgenden beschrieben und in der nachfolgenden Tabelle zusammenfassend aufgelistet.

Der Parameter `asso_min_support` gibt den Minimal-Support einer Regeln und eines *frequent itemsets* und der Parameter `asso_min_confidence` die Minimal-Confidence einer Regel an. Diese beiden Parameter sind entscheidend für das Ergebnis der Assoziationsanalyse, da sie die Laufzeit des Algorithmus beeinflussen und zugleich die Menge der Assoziationsregeln regulieren. Der Parameter `asso_max_rule_length` gibt die maximale Länge der Assoziationsregeln an, d. h. Anzahl der Elemente in der Prämisse der Regel. Je größer dieser Wert gewählt wird, um so länger braucht der Algorithmus. Der Parameter `algo_name` für den Algorithmus zur Assoziationsanalyse muss nicht explizit angegeben werden, da nur dieser eine implementiert ist und sowieso verwendet wird.

Einstellung	Werte	Beschreibung
`algo_name`	`algo_apriori_association_rules`	Spezifiziert den Algorithmus zur Assoziationsanalyse
`asso_max_rule_length`	$x \in [2, 20], x = 4$	max. Anzahl an Items in der Assoziationsregel
`asso_min_support`	$x \in [0, 1], x = 0.1$	Minimum-Support für die Assoziationsregel
`asso_min_confidence`	$x \in [0, 1], x = 0.1$	Minimum-Confidence für die Assoziationsregel

Assoziationsanalyse mit Oracle Data Mining:

Es soll im Folgenden der Prozess des Assoziationsanalyse (siehe Abb. 3.6) exemplarisch demonstriert werden. In Listing 3.17 ist ein entsprechender Code-Ausschnitt gezeigt.

Bei diesem Beispiel soll nach Assoziationsregeln mit einem Minimum-Support von $0.2 = 20\%$ und einer Minimum-Confidence von $0.5 = 50\%$ gesucht werden. Die Einstellungen für die Assoziationsanalyse werden in der Konfigurationstabelle `ASSO_SETTINGS` gespeichert und anschließend der Methode `CREATE_MODEL` übergeben.

```
BEGIN
   INSERT INTO ASSO_SETTINGS (setting_name, setting_value)
       VALUES (DBMS_DATA_MINING.asso_min_support, 0.2);
   INSERT INTO ASSO_SETTINGS (setting_name, setting_value)
       VALUES (DBMS_DATA_MINING.asso_min_confidence, 0.5);
END;
```

```
/
-- Assoziationsanalyse über alle Attribute der Tabelle
DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'ASSO_MODEL',
    mining_function     => DBMS_DATA_MINING.association,
    data_table_name     => 'data',
    case_id_column_name => 'object_id',
    setting_table_name  => 'ASSO_SETTINGS');
```

Listing 3.17: Assoziationsanalyse

Im Anschluss ist das Ergebnis der Assoziationsanalyse – Assoziationsregeln und *frequent itemsets* – im Modell-Objekt mit dem Namen `ASSO_MODEL` gespeichert. Mit den Funktionen `GET_ASSOCIATION_RULES` für die Assoziationsregeln und `GET_FREQUENT_ITEMSETS` für die *frequent itemsets* können die Ergebnisse aus dem Modell-Objekt abgefragt werden.

Listing 3.18 zeigt die SQL-Anfrage, die die Ausgabe der Top-10 Frequent Itemsets der Assoziationsanalyse bewirkt. Da die Elemente der *frequent itemsets* geschachtelte Tabelle sind, ist die Ausgabe durch die SQL-Anfrage auf mehrere Zeilen verteilt, so dass die Zeilen mit derselben `ID` als Einheit betrachtet werden müssen.

```
-- Anfrage zur Ausgabe der Frequent Itemsets
SELECT DISTINCT F.itemset_id as id, I.column_value AS item,
       F.support*100 || '%', F.number_of_items
  FROM TABLE(DBMS_DATA_MINING.GET_FREQUENT_ITEMSETS(
                  'ASSO_MODEL',    -- Name des Modells
                           10      -- Top-N Mengen
             )) F, TABLE(F.items) I

ID   ITEM                    SUPPORT     NUMBER_OF_ITEMS
---  ---------------------- ----------- ---------------
 5   Attribut_A = Wert_1    12.41465%         4
 5   Attribut_B = Wert_3    12.41465%         4
 5   Attribut_C = Wert_8    12.41465%         4
 5   Attribut_D = Wert_2    12.41465%         4
                        ...
12   Attribut_A = Wert_1    3.724395%         3
12   Attribut_E = Wert_3    3.724395%         3
12   Attribut_D = Wert_4    3.724395%         3
                        ...
```

Listing 3.18: Ausgabe der Frequent Itemsets

Listing 3.19 zeigt die SQL-Anfrage, die die Ausgabe der Top-10 Assoziationsregeln bewirkt. Bei den Assoziationsregeln sind die Prämisse und die Konklusion der Regel als geschachtelte Tabelle gespeichert und wie bereits bei den Frequent Itemsets verteilt die SQL-Anfrage die Assoziationsregel über mehrere Zeilen. In Listing 3.19 sind demnach zwei Assoziationsregeln ($ID = 1003$ und $ID = 1004$) dargestellt, deren Prämisse jeweils aus drei Elementen besteht.

```
SELECT A.rule_id as id, A.rule_support*100 || '%' as sup,
       A.rule_confidence*100 || '%' as conf,
```

```
       B.attribute_name as ante, C.attribute_name as cons
 FROM TABLE(DBMS_DATA_MINING.GET_ASSOCIATION_RULES(
                'ASSO_MODEL', 10)) A,
      TABLE(A.antecedent) B, TABLE(A.consequent) C

ID     PRÄMISSE              KONKLUSION            SUP    CONF
-----  --------------------- --------------------  -----  -----
1003   Attribut_A = Wert_1   Attribut_C = Wert_8   0.12%   100%
1003   Attribut_B = Wert_3   Attribut_C = Wert_8   0.12%   100%
1003   Attribut_D = Wert_2   Attribut_C = Wert_8   0.12%   100%
1004   Attribut_A = Wert_1   Attribut_E = Wert_3   0.12%   100%
1004   Attribut_D = Wert_4   Attribut_E = Wert_3   0.12%   100%
1004   Attribut_D = Wert_3   Attribut_E = Wert_3   0.12%   100%
                       ...
```

Listing 3.19: Ausgabe der Assoziationsregeln

Klassifikation

In ODM wird die Klassifikation in drei Phasen unterteilt. In der ersten Phase erfolgt die Konstruktion des Modell-Objekts (Klassifikator), in der zweiten Phase erfolgt die Bewertung des Klassifikators und schließlich in der dritten Phase erfolgt die Klassifizierung von neuen unklassifizierten Daten unter Anwendung des Modells. Diese Einteilung unterscheidet sich geringfügig vom Vorgehen aus Kapitel 2, in der die Bewertung und die Klassifizierung zu einem Schritt zusammengefasst worden sind.

Die **erste Phase** ist in Abb. 3.8 schematisch illustriert und zeigt detailliert die einzelnen Teilschritte für die Konstruktion des Klassifikators. Der Teilschritt Spezifikation ist dabei zum einen optional in der Anwendung (und wird daher im Weiteren nicht näher betrachtet) und zum anderen ist die Angabe von Kosten, Prioritäten und Gewichten abhängig vom Klassifikations-Algorithmus. Durch die Angabe von Kosten, Prioritäten oder Gewichten kann/soll die Konstruktion des Klassifikators beeinflusst werden. Dies erfolgt beispielsweise durch die Gewichtung der Attribute mittels Prioritäten, um deren Relevanz hervorzuheben. [Ora06b].

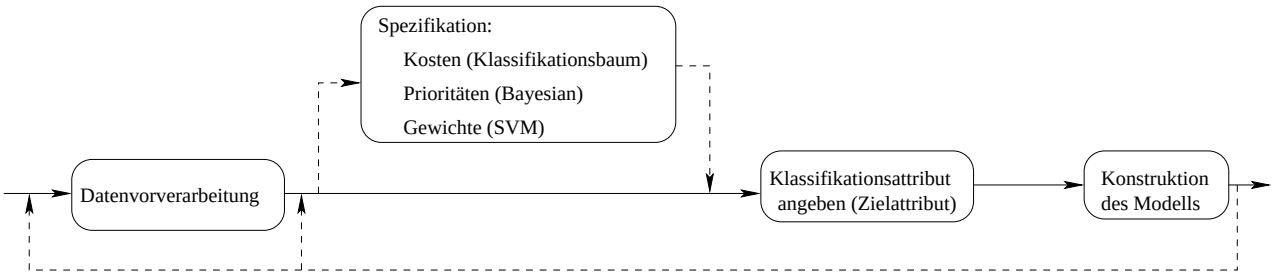


Abbildung 3.8: Konstruktion des Modells (Klassifikator)

In ODM werden für die Klassifikation gleich vier Algorithmen bereitgestellt, die im Folgenden beschrieben sind.

Klassifikationsbaum:

In ODM ist der Klassifikationsbaum als Algorithmus zur Klassifikation implementiert und unterstützt die Vorhersage (Klassifikation) von binären und mehrwertigen Attributen. Dabei werden jedoch nur binäre Aufteilungen (engl. splits) anhand eines Aufteilungskriteriums (Gini-Index und

Entropy – siehe Kapitel 2.6.4) unterstützt. Die Wahl des Aufteilungskriteriums wird hierbei über den Parameter `tree_impurity_metric` festgelegt. Diese Variante der Konstruktion hat den Vorteil, dass der Klassifikationsbaum nicht so stark in die Breite wächst und somit übersichtlich bleibt. Es hat jedoch auch den Nachteil, dass beim Aufteilen Attribut-Wiederholungen (=Attribut wird mehr als einmal zur Aufteilung einer Menge benutzt) vorkommen können. Die Höhe des Klassifikationsbaums wird auf der anderen Seite durch den Parameter `tree_term_max_length` gesteuert, damit dieser nicht unnötig hoch wird und schnelle Klassifizierungen möglich sind.

Die Konstruktion des Klassifikationsbaums erfolgt in ODM automatisch ohne Benutzer-Interaktionen. Der automatische Ablauf des Konstruktionsprozesses wird, wie bereits beschrieben, durch mehrere Parameter beeinflusst, die versuchen einen optimalen Klassifikationsbaum zu erstellen, der weder an over-fitting noch an under-fitting der Daten leidet. Weitere Parameter wie `tree_term_minpct_node`, `tree_term_minpct_split`, `tree_term_minrec_node` und `tree_term_minrec_split` beschreiben die Eigenschaften der Knoten im Klassifikationsbaum und beeinflussen gleichzeitig das Aufteilungskriterium und das Terminierungskriterium des Algorithmus.

Bei der Klassifikation mit dem Klassifikationsbaum sind des weiteren folgende Punkte zu beachten:

- Keine Unterstützung von geschachtelten Tabellen
- Keine Beeinflussung durch fehlende Werte (kein Preprocessing nötig)

Die Parameter für den Klassifikationsbaum sind der folgenden Tabelle zu entnehmen, wobei die Standardwerte **fett** hervorgehoben sind.

Parameter	Werte	Beschreibung
`algo_name`	`algo_decision_tree`	Spezifiziert Klassifikationsbaum als Klassifikations-Algorithmus
`tree_impurity_metric`	`tree_impurity_entropy`, **`tree_impurity_gini`**	Metriken zur Bestimmung des besten Splits in möglichst homogene Mengen
`tree_term_max_depth`	$x \in [2,20]$, $x = 7$	Aufteilungskriterium, bezeichnet die max. Höhe des Klassifikationsbaums
`tree_term_minpct_node`	$x \in [1,10]$, $x = 0.05$	Kein Nachfolgerknoten soll weniger als die angegebene Anzahl an Objekten, die dem Prozentsatz der Trainingsmenge entspricht, haben.
`tree_term_minpct_split`	$x \in [0,20]$, $x = 0.1$	Aufteilungskriterium, das die minimale Anzahl an Objekten, dargestellt als Prozentsatz der Trainingsmenge, in einem Vorgängerknoten, bezeichnet. Keine Aufteilung, falls die Anzahl geringer als der angegebene Wert ist.
`tree_term_minrec_node`	$x \in [0,\infty]$, $x = 10$	Kein Nachfolgerknoten soll weniger als die angegeben Anzahl an Objekten haben.
`tree_term_minrec_split`	$x \in [0,\infty]$, $x = 20$	Aufteilungskriterium, das die minimale Anzahl an Objekten im Vorgängerknoten, ausgedrückt als Zahl, bezeichnet.

Naive Bayes-Klassifikation:

Der Naive Bayes-Algorithmus zur Klassifikation (siehe Kapitel 2.6.4) ist ebenfalls in ODM implementiert und unterstützt ebenso die Klassifikation von binären und mehrwertigen Attributen.

Die Konstruktion des Klassifikators mit dem Naive Bayes-Algorithmus wird ausschließlich über die beiden Parameter `nabs_singleton_threshold` und `nabs_pairwise_threshold` gesteuert. Diese Parameter sind sogenannte Filter, die zur Filterung von kleinen Wahrscheinlichkeiten, im Bezug auf die Attributwerte, dienen, z. B. beschreibt der Parameter `nabs_singleton_threshold` = 0.03, das der Wert für ein Attribut in mind. 3% aller Datensätze vorkommen muss [Sch06]. Und analog dazu beschreibt der Parameter `nabs_pairwise_threshold` = 0.03, das in diesem Fall zwei Werte von Attributen in mind. 3% aller Datensätze vorkommen müssen.

Bei der Verwendung des Naive Bayes-Algorithmus zur Klassifikation gibt Oracle folgende Empfehlungen, um möglichst gute Resultate zu erzielen [Ora06d]:

- numerische und nicht-numerische Attribute mit **Equi-height Binning** diskretisieren
- Behandlung von Ausreißern

Die Parameter für den Naive Bayes-Algorithmus sind der folgenden Tabelle zu entnehmen.

Parameter	Werte	Beschreibung
`algo_name`	`algo_naive_bayes`	Spezifiziert Naive Bayesian als Klassifikations-Algorithmus
`nabs_singleton_threshold`	$x \in [0,1]$, $x = 0.01$	Es sollen nur die Werte von Attributen berücksichtigt werden, die diesen Grenzwert überschreiten.
`nabs_pairwise_threshold`	$x \in [0,1]$, $x = 0.01$	Das gleiche wie Single Threshold, jedoch gilt dieser Grenzwert für zwei Werte von Attributen.

Adaptive Bayesian Network-Klassifikation:

Bei dem Adaptive Bayesian Network-Algorithmus (ABN) handelt es sich einen proprietären Algorithmus von Oracle. Die Grundlage dieses Algorithmus basiert auf einem informationstheoretischen Ansatz (Minimum Description Length), der Attribute, sogenannte Prädikatoren, auswählt, anhand derer die Klassifikation durchgeführt wird. Diese Art der Klassifikation lehnt sich dabei an das Prinzip des Bayesschen Netzes (siehe Kapitel 2.6.4) an.

Das ABN-Modell besteht aus einer Menge von ***Network Features***, die durch den Minimum Description Length-Algorithmus ermittelt werden. Einfach ausgedrückt, handelt es sich beim *Network Feature* um eine Baumstruktur, ähnlich dem Klassifikationsbaum, die sich aus einem oder mehreren Attributen zusammensetzt und mit bedingten Wahrscheinlichkeiten annotiert ist. Dieser Klassifikationsbaum stellt das Bayessches Netz dar. Durch die baumartige Struktur des Bayesschen Netzes ist gewährleistet, dass die Wahrscheinlichkeitsverteilungen exakt berechnet werden können und nicht nur approximiert werden [Cha91]. Außerdem werden alle Prädikatoren durch den Algorithmus diskretisiert, so dass das Modell möglichst effektiv und einfach ist, z. B. Alter in $0-25$, $25-50$ und $50+$. Der ABN-Algorithmus besitzt drei Modi, mit denen er arbeiten kann:

1. Der erste Modus ist der **Naive Bayesian Build**. In diesem Modus wird ein Naive Bayesian-Modell (siehe *Naive Bayesian-Klassifikation*) mit Feature Selection (*Attribute Importance* – siehe Kapitel 3.5.2) erzeugt. Dabei werden die k Prädikatoren ausgewählt, die am meisten mit dem

Klassifikationsattribut korrelieren. Der Wert für k kann über den Parameter `abns_max_nb_predictors` angegeben werden und bezeichnet die max. Anzahl an Prädikatoren für die Klassifikation. Die Klassifikation erfolgt dann analog zur Naive Bayesian-Klassifikation, wobei die beiden Parameter automatisch auf 0.0 (Null) gesetzt werden [Sch06].

2. Der zweite Modus ist der **Single Feature Build**. In diesem Modus wird ein vereinfachter Klassifikationsbaum (Konstruktion nach dem C4.5-Algorithmus [Qui93]) erzeugt. Auf jeder Stufe des Baumes ist genau ein Prädikator, der so viele Nachfolger besitzt, wie der Prädikator disjunkte Werten. Die max. Anzahl an Prädikatoren, die zur Klassifikation verwendet werden sollen, wird über den Parameter `abns_max_predictors` angegeben. Bei diesem Modus werden zwei Ansätze – regelbasiertes Vorgehen und bedingte Wahrscheinlichkeiten (Bayes-Theorem) – miteinander kombiniert. Durch diese Kombination kann die Genauigkeit von Bayes ausgenutzt und gleichzeitig die Klassifikation durch das regelbasierte Vorgehen (Klassifikationsregeln) nachvollzogen werden.
3. Der dritte Modus ist der **Multi Feature Build**. In diesem Modus werden mehrere bedingt unabhängige *Network Features* erzeugt und derart behandelt als wären sie bedingt unabhängige Prädikatoren in einem Naive Bayes-Modell. Durch diese Kombination der *Network Features* wird die Genauigkeit der Klassifizierung erhöht. Auch bei diesem Modus wird die max. Anzahl an Prädikatoren, die zur Klassifikation verwendet werden sollen, über den Parameter `abns_max_predictors` angegeben.

Bei der Klassifikation mit dem Adaptive Bayesian Network-Algorithmus ist folgendes zu beachten:

- Keine Unterstützung von geschachtelten Tabellen
- Nur im Single Feature Build werden Klassifikationsregeln erzeugt
- numerische und nicht-numerische Attribute diskretisieren
- Behandlung von Ausreißern verbessert das Ergebnis

Die Parameter für den Adaptive Bayesian Network-Algorithmus sind der folgenden Tabelle zu entnehmen, wobei die Standardwerte **fett** hervorgehoben sind.

Parameter	Werte	Beschreibung
`algo_name`	`algo_adaptive_bayes_network`	Spezifiziert Adaptive Bayesian Network als Klassifikations-Algorithmus
`abns_model_type`	`abns_single_feature`, `abns_naive_bayes`, **`abns_multi_feature`**	Bezeichnet das Modell, den der Adaptive Naive Bayesian-Algorithmus verwenden soll
`abns_max_build_minutes`	$x \in [0, \infty]$, $x = 0$	max. Zeitlimit für die Erzeugung des Modells ($x = 0$ bedeutet kein Limit)
`abns_max_nb_predictors`	$x \in [0, \infty]$, $x = 10$	max. Anzahl von Prädiktoren zur Klassifizierung (Modus 1)
`abns_max_predictors`	$x \in [0, \infty]$, $x = 25$	max. Anzahl von Prädiktoren zur Klassifizierung (Modus 2,3)

Support Vector Machines-Klassifikation:

In ODM ist der Support Vector Machine-Algorithmus zur Klassifikation implementiert und unterstützt die Klassifikation von binären und mehrwertigen Attributen[BM05]. Die Support Vector Machine implementiert zwei Kernel-Funktionen: 1. Linearer-Kernel und 2. Gauss-Kernel. Die Wahl des Kernel zur Klassifikation kann entweder manuell über den Parameter `svms_kernel_function` eingestellt, oder automatisch durch den Algorithmus (auf Basis von Statistiken) bestimmt werden.

Die Funktionsweise des Support Vector Machines-Algorithmus wird über einige Parameter gesteuert. Da die Parameter einen entscheidenden Einfluss auf die Qualität des Klassifikationsmodells haben, können die Parameter in ODM automatisch durch den Algorithmus bestimmt werden und somit für ein optimales Ergebnis sorgen. Die Bestimmung der Parameter erfolgt dabei auf Basis der Charakteristiken der Daten. Eine kurze Beschreibung der wichtigsten Parameter wird in der folgenden Auflistung gegeben. Für eine detaillierte Beschreibung und deren Berechnung wird auf [BM05] verwiesen:

- Der Parameter `svms_conv_tolerance` steuert das Konvergenzverhalten des Lösungsalgorithmus, der intern vom Support Vector Machine-Algorithmus gelöst wird. Kleine Werte für diesen Parameter führen zu längeren Laufzeiten des Konstruktionsprozesses, aber auch zu genaueren Ergebnissen.
- Der Parameter `svms_complexity_factor` steuert das Verhalten im Bezug auf die Anpassung des Modells an die Trainingsmenge. Er beschreibt die Komplexität der Definition der Hyperebene. Ein zu großer Wert für den Komplexitätsfaktor führt bei *schlecht* trennbaren Mengen zu Overfitting des Modells. Ein zu kleiner Wert für den Komplexitätsfaktor führt zu Underfitting.
- Bei der Verwendung des Gauss-Kernel sind die Parameter `svms_std_dev` und `svms_kernel_cache_size` von Bedeutung. Der Parameter `svms_std_dev` bestimmt zusammen mit dem Komplexitätsfaktor das Anpassungsverhalten des Algorithmus an die Trainingsmenge, d. h. bei festem Komplexitätsfaktor führt ein zu hoher Wert für `svms_std_dev` zu Underfitting und ein zu kleiner Wert zu Overfitting.
- Ein Problem der Support Vector Machines ist, dass das Modell mit der Größe der Trainingsmenge wächst. Dies hätte zur Folge, dass ein Einsatz von Support Vector Machines für große Datenmenge unpraktisch wäre. Aus diesem Grund ist in ODM die Funktion *Aktives Lernen* integriert, das den Umgang mit großen Datenmenge ermöglichen soll. Die Idee des *Aktive Lernens* ist es, die Erzeugung des Modells auf die aussagekräftigsten Trainingsdaten zu reduzieren und somit die Größe des Modells klein zu halten. Als Abbruchbedingungen dienen dem *Aktiven Lernen* entweder die max. Anzahl an Support Vektoren oder die Messung, dass keine signifikante Verbesserung der Genauigkeit des Modells erzielt werden kann. Die Genauigkeit des Modell ist annähernd gleich mit und ohne *Aktivem Lernen* [Ora06d]. Das *Aktive Lernen* wird über den Parameter `svms_active_learning` gesteuert.

Neben Support Vector Machines ist in ODM die One-Class Support Vector Machine integriert, die einen Spezialfall darstellt und lediglich eine einzige Kategorie (Klassifikationsklasse) benötigt. Um die One-Class Support Vector Machine in ODM zu benutzten, wird das Klassifikationsattribut (`target_column_name`) im Konstruktionsprozess des Modells auf `NULL` gesetzt. Der One-Class Support Vector Machine-Algorithmus lernt zwischen den bekannten Daten, die zur Kategorie gehören, und den restlichen Daten zu unterscheiden. Das Ziel ist die Konstruktion einer Funktion, die entscheidet, ob etwas zur Kategorie gehört oder nicht. Die One-Class Support Vector Machine findet zum einen Anwendung bei der Identifikation von Ausreißern (siehe Preprocessing in diesem Kapitel) und zum anderen bei Problemen, bei denen die Daten von einer Kategorie einfach und die Daten von einer anderen Kategorie schwierig zu klassifizieren sind.

Anmerkung: Der Parameter `svms_outlier_rate` legt bei der One-Class Support Vector Machine eine sogenannte Ausreißerrate fest, die die Komplexität der Funktion beeinflusst.

Bei der Verwendung des Support Vector Machines-Algorithmus gibt Oracle folgende Empfehlungen, um möglichst gute Resultate zu erzielen:

- numerischen Attribute normalisieren
- Attribut-Reduktion durch Attribute Importance oder Feature-Extraction
- Behandlung von Ausreißer

Die Parameter für den Support Vector Machines-Algorithmus sind der folgenden Tabelle zu entnehmen, wobei die Standardwerte **fett** hervorgehoben sind.

Parameter	Werte	Beschreibung
`algo_name`	`algo_support_vector_machines`	Spezifiziert den Support Vector Machines-Algorithmus, den das Modell verwenden soll
`svms_active_learning`	`svms_al_disable`, **svms_al_enable**	Gibt an, ob aktives Lernen aktiviert ist. Aktives Lernen erzeugt ein reduziertes Model, sonst ein Standard-Model
`svms_kernel_function`	**svms_linear**, `svms_gaussian`	Kernel für die Support- Vector--Machine
`svms_kernel_cache_size`	$x \in [0, \infty]$, $x = 50000000$ Bytes	Cache-Größe für den Kernel (**Nur:** `svms_gaussian`)
`svms_conv_tolerance`	$x \in [0, \infty]$, $x = 0.001$	Konvergenz Toleranz für den Algorithmus
`svms_std_dev`	$x \in [0, \infty]$ und Default wird durch Algorithmus bestimmt	Standardabweichung für Algorithmus (**Nur:** `svms_gaussian`)
`svms_complexity_factor`	$x \in [0, \infty]$ und Default wird durch Algorithmus bestimmt	Wert für den Komplexitätsfaktor des Algorithmus
`svms_epsilon`	$x \in [0, \infty]$ und Default wird durch Algorithmus bestimmt	Wert für den Epsilonfaktor des Algorithmus
`svms_outlier_rate`	$x \in [0, 1]$, $x = 0.1$	Ausreißerrate in der Trainingsmenge. Kann nicht mit Komplexitätsfaktor verwendet werden. **Nur:** One-Class SVM.

Klassifikation mit Oracle Data Mining:

Es soll im Folgenden der Prozess der Konstruktion des Klassifikators (siehe Abb. 3.8) exemplarisch beschrieben werden. In Listing 3.20 wird dazu ein Code-Ausschnitt gezeigt, dass den Klassifikator erzeugt.

Die Klassifizierung soll mit dem Support Vector Machine-Algorithmus durchgeführt werden. Die einzigen Parameter, die in der Konfigurationstabelle `CLASS_SETTINGS` gespeichert werden, sind der Name des Algorithmus und die Deaktivierung von *Aktives Lernen*. Alle anderen Parameter für die Klassifikation mit Support Vector Machine sollen vom Algorithmus automatisch bestimmt werden.

Bei der Klassifikation in ODM ist das Vorgehen bei der Konstruktion des Klassifikators für alle Algorithmen gleich, wobei jeweils der Name des bevorzugten Algorithmus angegeben und die entsprechenden Parameter für den Algorithmus gesetzt werden müssen.

```
BEGIN
    INSERT INTO CLASS_SETTINGS (setting_name, setting_value)
         VALUES (DBMS_DATA_MINING.algo_name,
                 DBMS_DATA_MINING.algo_support_vector_machine);
    INSERT INTO CLASS_SETTINGS (setting_name, setting_value)
         VALUES (DBMS_DATA_MINING.svms_active_learning,
                 DBMS_DATA_MINING.svms_al_disable);
END;
```

```
/
DBMS_DATA_MINING.CREATE_MODEL(
     model_name          => 'CLASSIFICATION_MODEL',
     mining_function     => DBMS_DATA_MINING.classification,
     data_table_name     => 'data',
     case_id_column_name => 'object_id',
     target_column_name  => 'target_class',
     setting_table_name  => 'CLASS_SETTINGS');
```

Listing 3.20: Klassifikation mit Support Vector Machine

Informationen zum Klassifikations-Modell-Objekt:

Der Klassifikator ist in ODM im Modell-Objekt gekapselt. Für eine Klassifizierung muss entsprechend das Modell-Objekt – Klassifikator – eingesetzt werden.

Das Modell-Objekt beinhaltet – neben dem Klassifikator – ebenfalls Informationen, die den Konstruktionsprozess betreffen, z. B. Informationen über Einstellungen des Algorithmus und verwendete Attribute (Prädikatoren) bei der Klassifikation. Die bereitgestellten Informationen unterscheiden sich jedoch erheblich, so dass bei der einen Methode die Klassifizierung transparent und nachvollziehbar ist und bei einer anderen Methode die Klassifizierung im Modell-Objekt gekapselt wird, wodurch keine detaillierten Information zur Klassifizierung gegeben werden kann.

Für die einzelnen Algorithmen der Klassifikation stehen die folgenden Funktionen zur Verfügung, um detaillierte Informationen zum Prozess der Klassifikation und somit zum Klassifikator zu ermitteln:

- Klassifikationsbaum – `GET_MODEL_DETAILS_XML`
- Naive Bayes-Klassifikation – `GET_MODEL_DETAILS_NB`
- Adaptive Bayesian Network-Klassifikation – `GET_MODEL_DETAILS_ABN`
- Support Vector Machine-Klassifikation – `GET_MODEL_DETAILS_SVM`

Es soll im Folgenden eine kurze Beschreibung der `GET_MODEL_DETAILS`-Funktionen und der zur Verfügung stehenden Information bezüglich jedes Klassifikationsmodells gegeben werden:

- `GET_MODEL_DETAILS_SVM`: Diese Methode liefert lediglich Informationen, wenn die Konstruktion des Klassifikators mit dem lineare Kernel durchgeführt wird. Die Methode liefert eine Menge von *Linearen Koeffizienten*, repräsentiert durch Attribut-Koeffizient-Paare, die die n-dimensionale Hyper-Ebenen beschreiben. Durch die Attribut-Koeffizient-Paare lassen sich die einzelnen Klassifikationsbereiche unterteilen und die Klassifikation wird nachvollziehbar. Unter der Verwendung des Gauss-Kernel – bei der Konstruktion des Klassifikators – stehen hingegen keine Details zur Verfügung. Die Klassifizierung erfolgt in diesem Fall nach dem Black-Box-Prinzip.
- `GET_MODEL_DETAILS_XML`: Diese Methode liefert den Klassifikationsbaum in Form einer XML-Struktur. Die Spezifikation der XML-Struktur für den Klassifikationsbaum richtet sich dabei nach der Data Mining Group Predictive Model Markup Language (PMML).
- `GET_MODEL_DETAILS_NB`: Diese Methode liefert für jede Ausprägung (Wert) des Klassifikationsattributs eine Menge von Attribut-Wert-Paaren, die die jeweilige Klassifikationsklasse, d. h. einen Wert des Klassifikationsattributs, beschreiben. Zu den Informationen gehören das Klassifikationsattribut und alle Ausprägungen des Attributs, inklusive Wahrscheinlichkeiten, d. h. die disjunkten Werte und deren Auftrittswahrscheinlichkeiten. Außerdem wird zu jedem Wert des Klassifikationsattributs eine Menge von Attributen mit bedingten Wahrscheinlichkeiten gespeichert.

Die Berechnung der Wahrscheinlichkeiten basiert auf dem im Kapitel 2.6.4 beschriebenen Vorgehen. Die Wahrscheinlichkeiten des Klassifikationsattributs und die Wahrscheinlichkeiten der Attribute, die das Klassifikationsattribut beschreiben, summieren sich jeweils allesamt zur Wahrscheinlichkeit 1 (logisch) auf. Das Besondere ist, dass bei einer bestimmten Ausprägung eines Datensatzes berechnet werden kann, mit welcher Wahrscheinlichkeit dieser den jeweiligen Klassifikationsklassen angehört. Diese Informationen sind im Modell-Objekt für die Klassifikation mit dem Naive Bayes-Algorithmus gespeichert.

- `GET_MODEL_DETAILS_ABN`: Diese Methode liefert lediglich Informationen über den Klassifikator, wenn dieser als Parameter `Single Feature` verwendet worden ist. In diesem Fall handelt es sich, wie bei der Assoziationsanalyse um Regeln. Dies sind dann die sogenannten Klassifikationsregeln, die die Klassifizierung transparent machen. In den anderen Fällen, also einem anderer Modus bei der Konstruktion des Modells, werden keine Informationen über die Klassifizierung und den Klassifikator geliefert. Die Klassifizierung erfolgt nach dem Black-Box-Prinzip.

Die **zweite Phase** der Klassifikation, die Bewertungsphase, ist in ODM optional. Wenn eine Bewertung des Klassifikators durchgeführt wird, müssen die Daten, die zur Bewertung des Modells herangezogen werden, auf die gleiche Weise wie in der ersten Phase vorverarbeitet werden. Damit wird erreicht, dass keine Abweichungen oder Verfälschungen auftreten können. In Abb. 3.9 ist die zweite Phase detailliert illustriert.

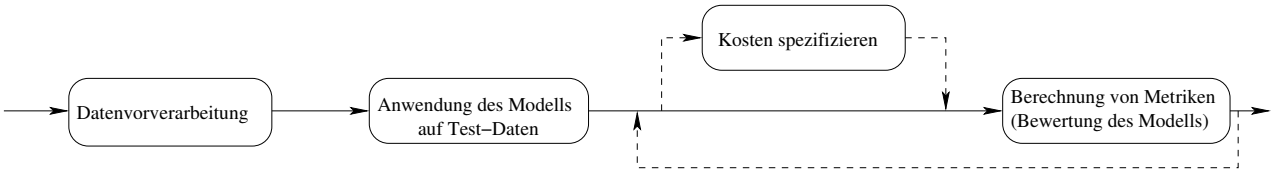


Abbildung 3.9: Bewertung des Klassifikators

Anmerkung: Auf den optionalen Teilschritt *Kosten spezifizieren* in Abb. 3.9 wird nicht weiter eingegangen. Für detaillierte Informationen zur Verwendung sei auf die Dokumentation von Oracle verwiesen [Ora06f].

Für die Bewertung der Klassifikationsmodelle sind in ODM die Confusion-Matrix, Lift und die ROC (Receiver Operating Characteristics) (siehe Kapitel 2.6.4) als Metriken implementiert. Da ausschließlich die Confusion-Matrix sowohl für die binäre als auch die mehrwertige Klassifikation eingesetzt werden kann, beschränkt sich die Beschreibung der Metriken lediglich auf die Confusion-Matrix. Für die Verwendung von Lift und ROC zur Bewertung von Klassifikationsmodellen sei auf die Literatur verwiesen (siehe Kapitel 2.6.4).

Confusion-Matrix:

In Listing 3.21 ist eine SQL-Anfrage enthalten, die eine Confusion-Matrix für ein Klassifikationsmodell unter Verwendung der SQL-Funktion `PREDICTION` generiert. In diesem Beispiel erfolgt die Klassifizierung in drei Gruppen, die durch die Werte `Attribut_Wert_1`, `Attribut_Wert_2` und `Attribut_Wert_3` repräsentiert sind. Die Genauigkeit des Modells lässt sich anschließend aus der Confusion-Matrix berechnen. Die SQL-Anfrage im Anschluss an die Confusion-Matrix demonstriert diese Berechnung.

```
SELECT target_attribute AS actual_target_value,
       PREDICTION(class_model USING *) AS predicted_target_value,
       COUNT(*) AS Anzahl
FROM test_data_table;
```

```
ACTUAL_TARGET_VALUE      PREDICTED_TARGET_VALUE   ANZAHL
------------------------ ----------------------   --------
Attribut_Wert_1          Attribut_Wert_1           14
Attribut_Wert_1          Attribut_Wert_3            2
Attribut_Wert_2          Attribut_Wert_2           18
Attribut_Wert_3          Attribut_Wert_1            4
Attribut_Wert_3          Attribut_Wert_3           13

SELECT SUM(DECODE(class, PREDICTION(class_model USING *), 1, 0))
       /count(*) as Accuracy
FROM test_data_table;

  ACCURACY
----------
       .88
```

Listing 3.21: Confusion-Matrix

Anmerkung: ODM stellt zur Berechnung der Confusion-Matrix auch eine explizite Funktion mit dem Namen `COMPUTE_CONFUSION_MATRIX` im Package `DBMS_DATA_MINING` zur Verfügung, die die Berechnung der obigen SQL-Anfragen kapselt. Es sei für die Anwendung dieser Funktion auf die Dokumentation [Ora06f] verwiesen.

In der **dritten Phase** wird die Klassifizierung von neuen Daten vorgenommen. Diese Daten müssen wiederum auf die gleiche Weise wie in der ersten und zweiten Phase vorverarbeitet werden, damit eine genaue und fehlerfreie Klassifizierung gewährleistet werden kann. In Abb. 3.10 ist der Prozess der Klassifizierung schematisch illustriert.

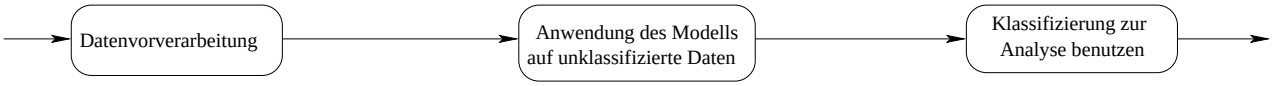


Abbildung 3.10: Anwendung des Models auf unklassifizierte Daten

Die Klassifizierung wird in ODM auf zwei unterschiedliche Arten durchgeführt. Die erste Möglichkeit besteht in der Verwendung der Methode `APPLY` im Package `DBMS_DATA_MINING` und die zweite Möglichkeit über die Verwendung der SQL-Funktionen `PREDICTION`, `PREDICTION_PROBABILITY` und `PREDICTION_SET` [Ora06h].

Die Klassifizierung mit der Methode `APPLY` stellt dabei die einfachere Möglichkeit dar. Die Methode bekommt als Parameter das Klassifikationsmodell und die Tabelle mit den unklassifizierten Daten. Das Ergebnis der Klassifizierung wird in einer Tabelle, dessen Name ebenfalls als Parameter übergeben wird, gespeichert. Das Schema dieser Tabelle ist für die Klassifikation folgendermaßen:

Spaltenname	Datentyp	Bedeutung
id	VARCHAR2/NUMBER	Identifier
prediction	VARCHAR2/NUMBER	Vorhersage/Klassifizierung
probability	NUMBER	Wahrscheinlichkeit der Vorhersage

In Listing 3.22 ist der Prozess der Klassifizierung unter Verwendung der Methode `APPLY` dargestellt. Die SQL-Anfrage im Anschluss gibt das Ergebnis der Klassifizierung aus. Die Ausgabe des Ergebnisses ist derart aufgebaut, dass für jede mögliche Klassifizierung (im Beispiel in drei Gruppen) die Wahrscheinlichkeit ausgegeben wird. In diesem Beispiel ist der Klassifikator sich seiner

Vorhersagen sehr sicher, weshalb lediglich '0' oder '1' als Wahrscheinlichkeitswerte auftreten (bei Unsicherheiten sind die Wahrscheinlichkeiten unterschiedlich verteilt).

```
DBMS_DATA_MINING.APPLY(
  model_name          => 'class_model',
  data_table_name     => 'apply_data',
  case_id_column_name => 'object_id',
  result_table_name   => 'apply_result');
END;
/

-- Ausgabe des Ergebnisses
SELECT id, prediction, probability FROM apply_result;

  ID                PREDICTION                PROBABILITY
  --                ---------------           -----------
   1                Attribut_Wert_1                   1
   1                Attribut_Wert_2                   0
   1                Attribut_Wert_3                   0
   2                Attribut_Wert_1                   0
   2                Attribut_Wert_2                   1
   2                Attribut_Wert_3                   0
                        ...
```

Listing 3.22: Klassifizierung unter Verwendung von APPLY

Anmerkung: Die Klassifizierung neuer Daten mit den SQL-Funktionen nimmt den Prozess der Klassifizierung mit der `APPLY`-Funktion auseinander. Es ist möglich die `APPLY`–Funktion zu simulieren, indem die Funktionen `PREDICTION` und `PREDICTION_PROBABILITY` verwendet werden (siehe Berechnung der Confusion-Matrix in Listing 3.21). Ein Vorteil der Verwendung dieser Funktionen besteht in der vielseitigen Einsetzbarkeit, wodurch verschiedene Ergebnisse erzielt werden können. Die Funktion `PREDICTION_SET` erlaubt beispielsweise die Analyse der Klassifikation durch eine Menge von möglichen Szenarien der Klassifizierung mit entsprechenden Wahrscheinlichkeiten. Für weitere Informationen und Anwendungen sei auf [Ora06f] verwiesen.

Regression

Die Regression läuft in Oracle Data Mining prinzipiell analog zur Klassifikation ab, jedoch können nicht alle Algorithmen, die bei der Klassifikation vorgestellt wurden, verwendet werden. In ODM kann die Regression lediglich mit dem Support Vector Machine-Algorithmus durchgeführt werden und das Klassifikationsattribut (`target_column_name`) muss einen der folgenden Datentypen besitzen:

- `INTEGER`, `FLOAT` oder `NUMBER`.

Des Weiteren ist für die Regression der Parameter `svms_epsilon` von besonderer Bedeutung, da dieser den ϵ-Bereich festlegt, in dem der Algorithmus eine Funktion zu finden versucht (siehe Kapitel 2.6.5). Die Angabe dieses Parameters ist jedoch nicht erforderlich, da dieser wiederum durch den Support Vector Machine-Algorithmus automatisch bestimmt wird. Damit soll das Ergebnis zu optimiert und die Interaktion zu minimiert werden.

Anmerkung: Die Parameter für den Support Vector Machine-Algorithmus im Fall der Regression sind der Tabelle aus dem Abschnitt Klassifikation 3.5.3 zu entnehmen.

In der **ersten Phase**, die analog zur ersten Phasen der Klassifikation in Abb. 3.8 ist, wird das Modell, also die Regressionsfunktion, für die Regression konstruiert. In Listing 3.23 ist die Konstruktion eines Regressionsmodells dargestellt, wobei der Parameter `target_column_name` numerisch sein muss.

```
DBMS_DATA_MINING.CREATE_MODEL(
     model_name          => 'REGRESSION_MODEL',
     mining_function     => DBMS_DATA_MINING.regression,
     data_table_name     => 'data',
     case_id_column_name => 'object_id',
     target_column_name  => 'target');
```

Listing 3.23: Regression

Das Modell für die Regression mit dem Support Vector Machine-Algorithmus kann bei ODM, wie bei der Klassifikation, nur eingesehen werden, wenn der Lineare Kernel benutzt wird. In Listing 3.24 ist die SQL-Anfrage abgebildet, die Details für die Regressionsfunktion ausgibt. Es werden die Koeffizienten für jedes Attribut, das an der Regressionsfunktion beteiligt ist wiedergegeben. Beispielsweise wurde ein Modell mit beliebigen Punkten der Funktion $f(x) = 4x - 20$ erstellt. Das Ergebnis, d. h. die Koeffizienten der Regressionsfunktion, der Regressionsanalyse ist in Listing 3.24 dargestellt.

```
SELECT B.attribute_name, B.attribute_value, B.coefficient
 FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_SVM('regression_model
    ')) A,
      TABLE(A.attribute_set) B;
ATTRIBUTE                      COEFFICIENT
------------------------------ -----------
X                              4.01894622
                               -20.521021
```

Listing 3.24: Details des Regressionsmodells

Die **zweite Phase** der Regression, die Bewertungsphase, dient analog zur Klassifikation zur Überprüfung der Genauigkeit des Modells. Die Methoden zur Messung der Genauigkeit, die für die Klassifikationsmodelle vorgestellt worden sind, können für die Regression nicht verwendet werden. Zur Messung der Genauigkeit für ein Regressionsmodell eignen sich die folgenden Maße:

- Root Mean square error (Wurzel des Mittleren-Quadratischen-Fehlers)
- Mean absolute error (Mittlerer-Absoluter-Fehler)

```
 -- 1. Root Mean Square Error - Sqrt(Mean((x - x')^2))
SELECT SQRT(AVG((A.prediction - B.target)
              * (A.prediction - B.target))) rmse
  FROM predicted_data A natural join test_data

RMSE
-------------
3.02
```

```
-- 2. Mean Absolute Error - Mean(|(x - x')|)
SELECT AVG(ABS(a.prediction - B.anz_movie)) mae
  FROM predicted_data A natural join test_data

MAE
-------------
2.56
```

Listing 3.25: Bewertung des Regressionsmodells

Die **dritte Phase** der Regression beschäftigt sich mit der eigentlichen Berechnung der neuen Daten unter Verwendung des Regressionsmodells. Das Vorgehen in dieser Phase entspricht dem der Klassifizierung in Abb. 3.10. Lediglich das Ergebnis der Regression unterscheidet sich von der Klassifikation. Die Tabelle, die die Vorhersage speichert, hat folgendes, um die Wahrscheinlichkeit reduziertes, Schema:

Spaltenname	Datentyp	Bedeutung
case_id	VARCHAR2/NUMBER	Identifier
prediction	NUMBER	Vorhersage

In Listing 3.26 ist die Vorhersage von Daten unter Verwendung des Regressionsmodells dargestellt. Die anschließende Ausgabe des Ergebnisses beschränkt sich auf die `id` des Objektes und den dazugehörigen Wert.

```
dbms_data_mining.apply(
  model_name          => 'regression_model',
  data_table_name     => 'apply_data',
  case_id_column_name => 'object_id',
  result_table_name   => 'apply_result');
END;
/

-- Ausgabe des Ergebnisses
SELECT case_id, prediction FROM apply_result;

ID             PREDICTION (Sollwert)
----           ---------------------
 2                  19.67  (20)
 3                  39.76  (40)
 4                  59.86  (60)
 5                  79.95  (80)
 6                 100.05 (100)
```

Listing 3.26: Klassifizierung unter Verwendung von APPLY

Anmerkung zur Regression:

Die mangelnde Unterstützung der Regression in der Oracle Data Mining-Cartridge ist dadurch zu begründen, dass die Regression im weitesten Sinne einer Klassifikation, mit einer entsprechenden Diskretisierung des Klassifikationsattributs, entspricht. Aus diesem Grund ist ebenfalls der Umfang zur Beschreibung der Regression auf diese knappe Erläuterung beschränkt.

4 Data Mining - Ein Beispiel

In diesem Kapitel soll das Data Mining, das zuvor theoretisch in Kapitel 2 und konkret anhand von Oracle Data Mining in Kapitel 3 eingeleitet wurde, an einer realen Datenbank untersucht werden. Die Datenbank, die verwendet werden soll, ist eine Film-Datenbank, im Folgenden Movie-Datenbank genannt, die in Abb. 4.1 in einem vereinfachten ER-Diagramm (Entity-Relationship-Diagramm) dargestellt ist.

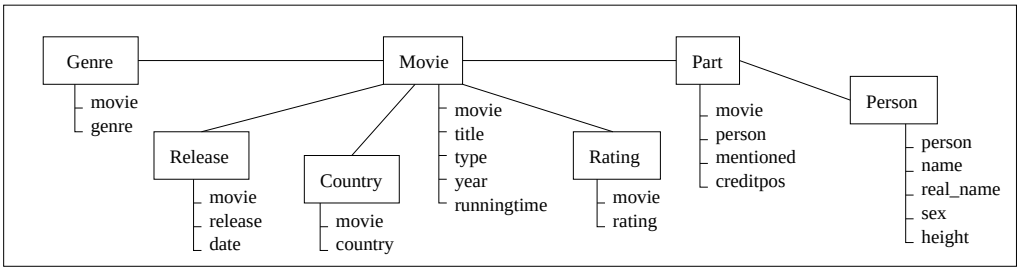


Abbildung 4.1: Movie-Datenbank

Bei der Movie-Datenbank handelt es sich um eine Sammlung von Film-Informationen, die sowohl Kino-Filme als auch Serien beinhaltet. Dabei repräsentiert der Film das zentrale Objekt (Film-Objekt), welches zusätzlich um zahlreiche Informationen angereichert ist. Diese Informationen wiederum verteilen sich auf verschiedene Tabellen, wobei jede Tabelle einen semantischen Bezug zum Film-Objekt hat. Die folgende Auflistung soll einen Überblick über die Daten und Tabellen geben, die in den nächsten Abschnitten von Bedeutung sind. Die Tabelle ...

- ... `Rating` gibt die Bewertung zu einem Film wieder. Außerdem ist darin die Anzahl der Stimmen, die zur Bewertung beitragen, enthalten.
- ... `Genre` gibt Auskunft, welchem Genre der Film zuzuordnen ist. Dabei kann ein Film auch mehreren Genres angehören, z. B. Film *X* gehört zum Genre *Crime* und *Thriller*.
- ... `Person` enthält eine Vielzahl an Informationen über eine Person, die in Filmen oder Serien mitgewirkt hat.
- ... `Part` repräsentiert dabei das Bindeglied zwischen dem Film und den Personen, die darin mitgewirkt haben, z. B. die Person *X* war Regisseur (Attribut `mentioned`) in Film *Y* oder war Schauspieler (Attribut `mentioned`) in Film *Z*.

Weitere Tabellen, die jedoch nicht näher erläutert werden sollen, enthalten beispielsweise Informationen über das Budget eines Films, Keywords (Schlagwort) zur Beschreibung eines Films oder Angaben über die Veröffentlichungen eines Films in verschiedenen Ländern.

In den folgenden Abschnitten werden die einzelnen Schritte des Knowledge Discovery, vom Preprocessing über die Transformation und das Data Mining bis zur einer einfachen Bewertung der Ergebnisse, anhand der Movie-Datenbank demonstriert. Dazu wird es jeweils eine Fragestellung bzgl. der Movie-Datenbank für jede Data Mining-Methode (siehe Kapitel 2) geben. Unter Berücksichtigung der Erkenntnisse und unter Anwendung der Werkzeuge aus Kapitel 3 soll versucht werden, die gestellten Fragestellungen zu beantworten. Neben der Vorgehensweise, werden auch Probleme und Erfahrungen, die sich bei der Knowledge Discovery ergeben haben, dokumentiert. Die Beispiele sind vollständig in Anhang B zu finden.

4.1 Assoziationsanalyse

Die Assoziationsanalyse bringt – einfach ausgedrückt – Beziehungen zum Vorschein, die in den Daten vorhanden sind. In der Movie-Datenbank werden beispielsweise Beziehungen zwischen den Schauspielern, die in einem Film mitgespielt haben, vermutet. Daher soll im Folgenden untersucht werden, ob es Schauspieler gibt, die mit gewissen anderen Schauspielern bevorzugt zusammen in Filmen mitgewirkt haben. Um dies letztlich herauszufinden, wird die Assoziationsanalyse verwendet. Das Vorgehen zur Lösung dieser Fragestellung wird dabei in die folgenden Schritte unterteilt (siehe Kapitel 2 – Ablauf der KDD):

1. *Bereitstellung der Daten*
2. *Preprocessing und Transformation*
3. *Data Mining – Assoziationsanalyse*
4. *Darstellung & Auswertung der Ergebnisse*

4.1.1 Assoziationsanalyse: `Schauspieler mit Schauspieler in Filmen`

1. Bereitstellung der Daten: Die Daten für die Assoziationsanalyse müssen zunächst zusammengefasst und eingeschränkt werden. Dazu werden alle Daten, die für die Analyse von Interesse sind, zu einer neuen Datenquelle (Tabelle bzw. Sicht) zusammengefasst. Danach werden die Daten derart reduziert und vorbereitet, dass Assoziationen vom Algorithmus gefunden werden können, sofern welche enthalten sind. Die bereitgestellten Daten über Schauspieler und Filme besitzen für diese Fragestellung folgende Einschränkungen:

- Schauspieler muss in mind. 20 Filmen mitgespielt haben.
- Film muss vom Typ `cinema` sein.
- Film wurde in den U.S.A. gedreht (sogenannter Hollywood-Film).
- Film stammt aus den Jahren zwischen 1950 und 2000.
- Im Film müssen mind. 2 Schauspieler (obige Einschränkung der Schauspieler) mitspielen.

2. Preprocessing und Transformation: Ein Teil des Preprocessing wurde bereits im ersten Schritt, bei der Bereitstellung der Daten, durchgeführt. Zum Preprocessing gehören nach Kapitel 2 u. a. die Behandlung von fehlenden Werten und von Ausreißern. Beides ist in diesem Fall nicht notwendig, da zum einen fehlende Werte, wenn diese auftreten, gewollt sind und zum anderen Ausreißer bei diesem Beispiel und dieser Data Mining-Methode nicht vorkommen können.

Eine Transformation mit den Methoden, wie sie in Kapitel 2 und 3 beschrieben wurden, ist bei diesem Beispiel der Assoziationsanalyse ebenfalls nicht notwendig. Denn eine Normierung der Attribute ist nur bei numerischen Werte durchführbar, die in diesem Beispiel nicht vorhanden sind. Eine Diskretisierung der Attribute kann zwar auch für nicht-numerische Attribute durchgeführt werden, ist aber in diesem Fall nicht sinnvoll. Auch eine Attribut-Reduktion muss nicht durchgeführt werden, da die Daten bereits genügend eingeschränkt sind.

Weiterhin ist noch zu bedenken, dass die bereitgestellten Daten in der Tabelle `asso_pre_prepared` als *Film-Schauspieler-Paar* gespeichert sind (Listing 4.1), wodurch die einzelnen Datensätze jedoch nicht eindeutig durch den Film identifiziert werden und somit eine Assoziationsanalyse nach der vorgegebenen Fragestellung, also welche Schauspieler gemeinsam in Filmen mitgespielt haben, nicht beantwortet werden kann. Aus diesem Grund ist es bei diesem Beispiel notwendig die Analogie zur Warenkorbanalyse anzuwenden. Dabei entspricht der Film dem Warenkorb und die Schauspieler den Produkten. Die dafür erforderliche Transformation der Tabelle wird dabei nach dem in Kapitel 3 vorgestellten Prinzip durchgeführt.

Die Transformation bewirkt, dass die Schauspieler als Attribute betrachtet werden, wobei ein Attribut jeweils mit dem Wert 1 markiert wird, wenn der Schauspieler in einem Film mitgespielt. Eine Markierung ist dabei obligatorisch, da die geschachtelte Tabelle wiederum ein Attribut-Wert-Paar darstellt und einen Wert erwartet, jedoch spielt sie bei der Assoziationsanalyse und den Assoziationsregeln später keine Rolle. Nach der Transformation sind die Datensätze jeweils durch den Film eindeutig identifizierbar.

```
-- einfache Definition der Tabelle ohne Einschränkungen
CREATE TABLE asso_pre_prepared AS
   SELECT movie as id, name as actor
   FROM movie natural join part natural join person;

CREATE TABLE asso_prepared
NESTED TABLE actor_for_movie STORE AS DM_Nested_Numericals_TABLE
AS (
    SELECT id,
        CAST(COLLECT(DM_Nested_Numerical(actor, 1))
             AS DM_Nested_Numericals) actor_for_movie
    FROM asso_pre_prepared
   );
```

Listing 4.1: Transformation der Daten

Im nächsten Schritt werden die Parameter für die Assoziationsanalyse bestimmt und festgelegt, also der Minimum-Support, die Minimum-Confidence und die maximale Länge der Regel. Zur Bestimmung der Parameter, speziell für die Länge der Regel und den Minimum-Support, werden die Daten analysiert. Listing 4.2 demonstriert eine einfache Analyse der Daten, in der zunächst die Anzahl der Filme und die Anzahl der Schauspieler ermittelt wird. Aus der Anzahl der Filme kann dann ein sinnvoller Wert für den Minimum-Support festgelegt werden, d. h. die Anzahl der Filme, in denen Schauspieler-Paarungen zusammen gespielt haben. Bei diesem Beispiel wird mit einem Minimum-Support von 1% und einer Minimum-Confidence von 10% begonnen und anschließend, falls nötig, korrigiert. Der Minimum-Support von 1% bedeutet in diesem Beispiel, dass eine Menge von Schauspielern zusammen in mind. 165 Filmen (1% von 16529 Filmen) gespielt haben müssen. Die Minimum-Confidence bedeutet, dass eine Regel, die aus der Menge der Schauspieler hervorgeht, zu 10% zutrifft. Im zweiten Schritt wird die durchschnittliche Anzahl an Schauspielern in Filmen berechnet. Das Ergebnis gibt Aufschluss über die zu erwartende Länge der Assoziationsregel. Diese kann durch einen Parameter beschränkt werden, was Auswirkung auf die Laufzeit der Analyse hat. Damit ist das Preprocessing und die Transformation abgeschlossen.

```
-- 1. Kardinalitäten von Movies und Schauspielern
SELECT count(id), count(distinct id), count(distinct actor)
FROM asso_pre_prepared;

Einträge  Filme   Schauspieler
--------- ------  -------------
156351    16529   12272

-- 2. Durchschnittliche Anzahl von Schauspielern pro Film
SELECT to_char(avg(actor), 99.99) avg_actor
FROM (select count(actor) actor from asso_pre_prepared group by id);
```

```
AVG_ACTOR
---------
 9.46
```

Listing 4.2: Analyse der Daten im Bezug auf die Parameter des Algorithmus

3. Data Mining – Assoziationsanalyse: Im folgenden Schritt wird das Modell mit den erforderlichen Einstellungen für die Assoziationsanalyse erstellt. Da die zuvor festgelegten Werte für die Parameter keine Ergebnisse in Form von Assoziationsregeln lieferten, sind diese in Listing 4.3 bereits angepasst. Die Ermittlung der Werte erfolgte dabei durch „Herumprobieren“, wobei untersucht wurde, ab welchem Support und welcher Confidence Assoziationsregeln gefunden werden (d. h. Support reduzieren und Confidence anpassen). Es ergeben sich für die Parameter folgende Werte: Minimum-Support liegt bei mind. 0.2% (also in mind. 33 von 16529 Filmen), Minimum-Confidence liegt bei mind. 50% und die max. Länge der Regel ist auf 6 festgelegt. Mit diesen Einstellungen findet die Assoziationsanalyse 12 Assoziationsregeln (Auszug der Regeln siehe Listing 4.4).

```
BEGIN             -- Einstellungen festlegen
    INSERT INTO test_asso_settings
           VALUES (dbms_data_mining.asso_min_support,0.002); -- 0.2%
    INSERT INTO test_asso_settings
           VALUES (dbms_data_mining.asso_min_confidence,0.5);-- 50%
    INSERT INTO test_asso_settings
           VALUES (dbms_data_mining.asso_max_rule_length,6);
END;

BEGIN             -- Modell erstellen
    DBMS_DATA_MINING.CREATE_MODEL(
       model_name              => 'test_asso_1',
       mining_function         => DBMS_DATA_MINING.ASSOCIATION,
       data_table_name         => 'asso_prepared',
       case_id_column_name     => 'id',
       settings_table_name     => 'test_asso_settings');
END;
```

Listing 4.3: Konstruktion des Modells

4. Darstellung & Auswertung der Ergebnisse: Das Ergebnis der Assoziationsanalyse ist im Modell-Objekt gespeichert und gibt Auskunft, welche Schauspieler häufig zusammen in Filmen gespielt haben. Aus den Regeln lässt sich anschließend beispielsweise folgern, dass die Schauspieler befreundet sind/waren oder das die Filme mit den Paarungen erfolgreich sind/waren. Die Assoziationsregeln können über die Funktion `GET_ASSOCIATION_RULES` abgerufen werden. In Listing 4.4 wird diese Funktion benutzt, um alle Assoziationsregeln auszugeben, die einen Minimum-Support von 0.2%, eine Minimum-Confidence von 60% aufweisen und dessen Länge der Regel mind. 2 ist. Die Regeln beschreiben also, wenn ein Schauspieler A in einem Film C mitgespielt hat, dann hat zu x% der Schauspieler B ebenfalls mitgespielt. Für weitere Informationen zur Methode sei auf die Dokumentation [Ora06f] verwiesen.

```
SELECT rule_id, rule_support*100 || '%' as support,
       rule_confidence*100 || '%' as confidence,
      B.attribute_name || B.conditional_operator ||
           B.attribute_str_value as antecedent,
```

```
      C.attribute_name || C.conditional_operator ||
           C.attribute_str_value as consequence
FROM TABLE(DBMS_DATA_MINING.GET_ASSOCIATION_RULES(
                 'test_asso_1',    -- Name des Modells
                  NULL, NULL,
                  0.6,             -- Minimum-Confidence=60%
                  0.002,           -- Minimum-Support=0.2%
                  NULL,            -- max. Länge der Regel
                  2                -- min. Länge der Regel
       )) A, TABLE(A.antecedent) B, TABLE(A.consequent) C
ORDER BY 2 desc, 3 desc

-- [Auszug: Assoziationsregel mit Schauspielern (=1 ist Markierung)]
RULE_ID  SUPPORT   CONFIDENCE   ANTECEDENT         CONSEQUENCE
-------  -------   ----------   --------------     ----------------
   2       .57%     100%       Fine, Larry= 1     Howard, Moe= 1
  10       .36%     100%       Howard, Shemp= 1   Fine, Larry= 1
  10       .36%     100%       Howard, Moe= 1     Fine, Larry= 1
  11       .21%     100%       Sitka, Emil= 1     Howard, Moe= 1
  11       .21%     100%       Fine, Larry= 1     Howard, Moe= 1
  12       .21%     100%       Sitka, Emil= 1     Fine, Larry= 1
  12       .21%     100%       Howard, Moe= 1     Fine, Larry= 1
   7       .36%      98.36%    Howard, Shemp= 1   Howard, Moe= 1
   5       .36%      98.36%    Howard, Shemp= 1   Fine, Larry= 1
   3       .57%      96.94%    Howard, Moe= 1     Fine, Larry= 1
   8       .36%      63.16%    Howard, Moe= 1     Howard, Shemp= 1
   8       .36%      63.16%    Fine, Larry= 1     Howard, Shemp= 1
   6       .36%      61.22%    Howard, Moe= 1     Howard, Shemp= 1
```

Listing 4.4: Auszug: Ausgabe der Assoziationsregeln

Die Assoziationsregeln in Listing 4.4 sind schlecht formatiert und somit recht unleserlich. Die Schreibweise für Assoziationsregeln, wie sie in Kapitel 2 beschrieben ist, macht die obigen Regeln für den Benutzer verständlicher, z.B. die Regel mit der ID=10:

`Howard, Shemp AND Howard, Moe` $\Rightarrow$ `Fine, Larry` [s=0.36%, c=100%]

Die *frequent itemsets* können mit der Funktion `GET_FREQUENT_ITEMSETS` abgerufen werden. Die Verwendung der Funktion erfolgt analog zur Funktion für die Assoziationsregeln. In Listing 4.5 ist die Ausgabe der *frequent itemsets* (mit mind. zwei Elementen) dargestellt, sie unterscheidet sich nicht gravierend von der der Assoziationsregeln.

```
SELECT distinct id, item, support, number_of_items
 FROM (SELECT itemset_id as id, I.column_value AS item,
              F.support*100 as support, -- Support
              F.number_of_items         -- Anzahl der Elemente
         FROM TABLE(DBMS_DATA_MINING.GET_FREQUENT_ITEMSETS(
                    'test_asso_1')) F,
              TABLE(F.items) I
        where number_of_items > 1)
ORDER BY id;
```

```
ID    ITEM                        SUPPORT    NUMBER_OF_ITEMS
----- ----------------------   ---------- ----------------
 999   Blanc, Mel-480893          .23              2
 999   Foray, June-121279         .23              2
1000   Farnum, Franklyn-610072    .22              2
1000   Flowers, Bess-120373       .22              2
1001   Fine, Larry-616985         .57              2
1001   Howard, Moe-702359         .57              2
                           ...
1006   Fine, Larry-616985         .36              3
1006   Howard, Moe-702359         .36              3
1006   Howard, Shemp-702427       .36              3
                           ...
```

Listing 4.5: Ausgabe der *frequent itemsets*

Diese Funktion, d. h. die Ausgabe der *frequent itemsets*, wird genau dann interessant, wenn die Assoziationsanalyse keine Regeln gefunden hat. In diesem Fall werden die *frequent itemsets* untersucht und ermittelt, ob es überhaupt welche gibt, die mehr als zwei Elemente haben, woraus Assoziationsregeln gebildet werden können. Ist dies der Fall, kann die Minimum-Confidence für die Assoziationsregeln entsprechend reduziert werden. Tritt jedoch der Fall ein, dass keine *frequent itemsets* oder lediglich *frequent itemsets* mit genau einem Element vorhanden sind, dann muss der Minimum-Support reduziert werden.

4.1.2 Erfahrungen mit der Assoziationsanalyse in ODM

Ein Problem bei der Assoziationsanalyse stellt der große Speicherbedarf, den der Algorithmus bei der Analyse der Daten benötigt, dar. Der Algorithmus stoppt die Analyse mit der in Listing 4.6 abgebildeten Fehlermeldung. Die Ursache liegt darin, dass nicht genügend Speicher für die temporären Daten in der Datenbank (`tablespace`) vorhanden ist.

```
ERROR at line 1:
ORA-40101: Data Mining System Error ODM_ASSOCIATION_MODEL-BUILD
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 105
ORA-06512: at "DMSYS.ODM_ASSOCIATION_RULE_MODEL", line 1396
ORA-01653: unable to extend table Z10_32NKBT by 1024 in tablespace
```

Listing 4.6: Probleme bei Assoziationsanalyse

Bei Assoziationsanalyse von eindimenstionalen Assoziationsregeln und die damit verbundene Verwendung der geschachtelten Tabellen (`DM_NESTED_NUMERICALS` und `DM_NESTED_CATEGORICAL`) tritt ein weiteres Phänomen auf. Im obigen Beispiel wurde beispielsweise der Datentyp `DM_NESTED_NUMERICALS` verwendet und die Assoziationsanalyse dauerte ca. 15 sec. Bei der Verwendung von `DM_NESTED_CATEGORICAL` (Listing 4.7) hingegen benötigte die gleiche Analyse bereits ∼2 min 30 sec, war also ∼10 mal langsamer.

```
-- Die Markierung ist ein String '1' und keine Zahl 1
CAST(COLLECT(DM_Nested_Categorical(actor, '1'))
          AS DM_Nested_Categoricals) actor_for_movie
```

Listing 4.7: `DM_NESTED_NUMERICALS` vs. `DM_NESTED_CATEGORICAL`

4.2 Regression

In der Movie-Datenbank soll untersucht werden, ob es zwischen dem Jahr und der Anzahl produzierter Filme eine Abhängigkeit in Form einer Regressionsfunktion gibt. Diese Regressionsfunktion ist bewusst nur von einem Parameter abhängig, damit diese anschaulich in einem zweidimensionalen Diagramm dargestellt werden kann. Es wird also eine Funktion gesucht, die bei Angabe der Jahreszahl die Anzahl der produzierten Filme in dem Jahr berechnet.

Das Vorgehen dieser Data Mining-Aufgabe, der Regressionsanalyse, wird in die folgenden Schritte unterteilt:

1. *Bereitstellung der Daten*
2. *Preprocessing und Transformation*
3. *Data Mining – Regressionsanalyse*
4. *Testen des Regressionsmodells (Genauigkeit)*
5. *Darstellung & Auswertung der Ergebnisse*

4.2.1 Regressionsanalyse: `Jahr und Anzahl produzierter Filme pro Jahr`

1. Bereitstellung der Daten: Zunächst werden die Daten, die zur Definition der Regressionsfunktion erforderlich sind, zusammengefasst. In Listing 4.8 wird dazu eine Tabelle definiert, die die notwendigen Informationen – Identifikation der Datensätze (ID), Jahreszahl (YEAR) und Anzahl der produzierten Filme (ANZ_MOVIE) – enthält. Des weiteren sind die Daten auf die Jahre zwischen 1920 und 2005 beschränkt.

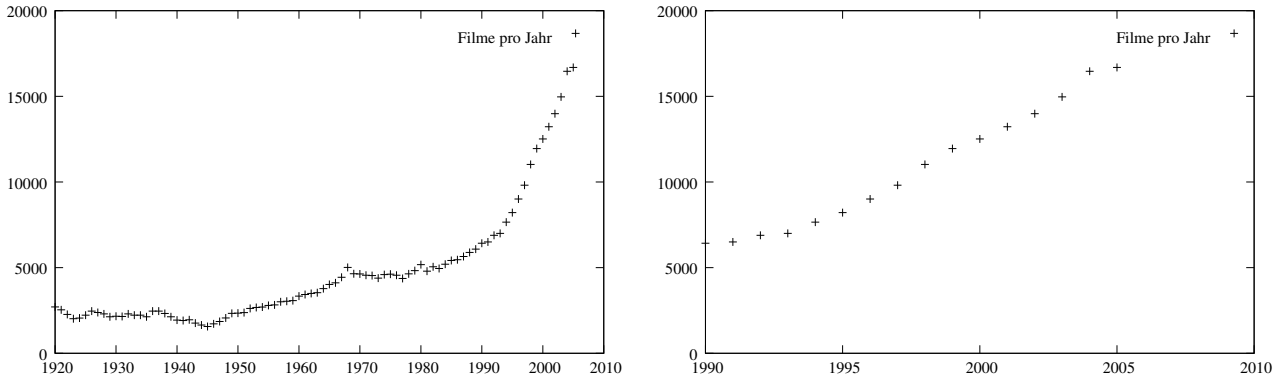
```
CREATE TABLE regression_data AS
  select rownum as id, year, anz_movie
    from (
       select year, count(a.movie) as anz_movie
         from moviedb.movie a
        where year >= 1920 and year <= 2005
         group by year)  f;
```

Listing 4.8: Daten zur Regressionsanalyse

In Abb. 4.2 (a) sind die Daten dieser Tabelle graphisch in einem Diagramm dargestellt. Aus dem Diagramm ist erkennbar, dass die Regressionsfunktion – zwischen den Jahren 1920 und 2005 – nicht linear verlaufen wird. Da die Regressionsanalyse in ODM mit dem Support Vector Maschine-Algorithmus (siehe Kapitel 3) durchgeführt wird, werden im nicht-linearen Fall keine Informationen über das Modell und somit auch keine Angaben über die Regressionsfunktion verfügbar sein. Aus diesem Grund wird der Bereich zur Definition der Regressionsfunktion im zweiten Schritt auf die Jahre zwischen 1990 und 2005 beschränkt, da die Regressionsfunktion in diesem Bereich vermutlich linear ist. Der eingeschränkte Bereich ist in Abb. 4.2 (b) graphisch dargestellt.

Anmerkung: Die Veranschaulichung des Problems mittels der Diagramme ist in diesem Beispiel gut realisierbar und vermutlich erscheint dadurch die Regressionsanalyse überflüssig, aber der Aspekt der Visualisierung geht bei mehr-dimensionalen Problemen schnell verloren.

2. Preprocessing und Transformation: In diesem Schritt wird das Preprocessing und die Transformation der Daten durchgeführt. Da die Daten jedoch derart stark selektiert wurden, dass keine fehlenden Werte enthalten sind, kann auf eine Behandlung von fehlenden Werte verzichtet werden.



(a) Filme pro Jahr von 1920 – 2005

(b) Filme pro Jahr von 1990 – 2005

Abbildung 4.2: Diagramm von Jahr und Anzahl produzierter Filme pro Jahr

Auch eine Behandlung der Ausreißer im Preprocessing-Schritt wird nicht durchgeführt, da sie bei der vorliegenden Kurve, die offensichtlich monoton steigend ist, keinen Sinn macht.

Im Transformationsschritt werden die Attribute `Jahr` und `Anzahl der Filme pro Jahr` normiert. Der Grund ist, dass der Algorithmus für die Regressionsanalyse in ODM von einer Normierung profitieren kann [Ora06d]. Als Normierungsfunktion wird die Min-Max-Normierung (siehe Kapitel 2.5) verwendet. Listing 4.9 demonstriert die Normierung der Daten.

```
BEGIN
        DBMS_DATA_MINING_TRANSFORM.CREATE_NORM_LIN('
           conf_reg_sample_1_1');
        DBMS_DATA_MINING_TRANSFORM.INSERT_NORM_LIN_MINMAX(
                'conf_reg_sample_1_1',
                'reg_data_sample_1',
                DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST(
                        'id')
        );
        DBMS_DATA_MINING_TRANSFORM.XFORM_NORM_LIN(
                'conf_reg_sample_1_1',  -- Parameter-Werte
                'reg_data_sample_1',    -- Datentabelle
                'tmp_reg_prepared'      -- Ergebnis
        );
END;
```

Listing 4.9: Normierung der Daten `Jahr` und `Anzahl der Filme pro Jahr`

Vor dem nächsten Schritt, dem Data Mining, werden die Daten in Trainingsdaten (`build_data`) und Testdaten (`test_data`) aufgeteilt. Die Trainingsdaten werden zur Konstruktion der Regressionsfunktion und die Testdaten zum anschließenden Testen der Genauigkeit der Funktion verwendet. Dabei erfolgt die Selektion der Testdaten stichprobenartig aus der gesamten Datenmenge (Stichprobe bedeutet in diesem Fall, dass jeder dritte Eintrag zum Testen verwendet wird).

3. Data Mining – Regressionsanalyse: In diesem Schritt wird das Modell, die Regressionsfunktion, ermittelt. Ausgehend von den Daten in Abb. 4.2 (a) und (b) wird die Regressionsanalyse zweimal durchgeführt.

Die erste Regressionsanalyse erhält die kompletten Daten aus Abb. 4.2 (a) und erstellt darauf basierend die Regressionsfunktion. Die zweite Regressionsanalyse erhält für die Konstruktion der Regressionsfunktion lediglich die auf den Bereich zwischen den Jahren 1990 und 2005 eingeschränkten Daten. Außerdem wird im zweiten Fall die Verwendung des linearen Kernels zur Konstruktion der Regressionsfunktion explizit angegeben. Alle weiteren Parameter sollen durch den Algorithmus bestimmt werden, da dadurch ein möglichst optimales Ergebnis erreicht wird. In Listing 4.10 wird die Regressionsanalyse demonstriert, wobei die Vorgabe des linearen Kernel nur für die zweite Regressionsanalyse gelten soll.

```
-- Vorgabe des linearen Kernel zur Konstruktion
BEGIN
  INSERT INTO regression_settings VALUES
      (DBMS_DATA_MINING.svms_kernel_function,
       DBMS_DATA_MINING.svms_linear);
END;

BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
     model_name              => 'regression_model',
     mining_function         => DBMS_DATA_MINING.REGRESSION,
     data_table_name         => 'build_data',  -- Training
     case_id_column_name     => 'id',
     target_column_name      => 'target',
     settings_table_name     => 'regression_settings' );
END;
```

Listing 4.10: Regressionsanalyse

4. Testen des Regressionsmodells: Nachdem die Regressionsfunktion (Modell) konstruiert worden ist, kann deren Genauigkeit mit den in Kapitel 3.5.3 vorgestellten Methoden gemessen werden. In Listing 4.11 wird die Genauigkeit mit den `Root Mean Square Error` und `Mean Absolute Error` gemessen, wobei zunächst die Werte für die Testdaten unter Anwendung des Modells, also der Regressionsfunktion, berechnet werden. Als Ergebnis ergeben sich für den `Root Mean Square Error` eine Abweichung von 926 Filmen und für den `Mean Absolute Error` eine Abweichung von 778 Filmen. Das ist, in Anbetracht der Dimension von `Anzahl der Filme`, eine gute Annäherung an die realen Daten durch die Regressionsfunktion.

```
BEGIN
  DBMS_DATA_MINING.APPLY(
       model_name              => 'regression_model',
       data_table_name         => 'test_data', -- Test
       case_id_column_name     => 'id',
       result_table_name       => 'test_reg_score'
  );
END;
-- 1. Root Mean Square Error - Sqrt(Mean((x - x')^2))
SELECT SQRT(AVG((A.prediction - B.target) *
                (A.prediction - B.target))) rmse
  FROM test_reg_score A natural join test_data B;
```

```
RMSE            |   Umrechnung in ANZ_MOVIE
---------       |   ----------------------
.09027          |   926

-- 2. Mean Absolute Error - Mean(|(x - x')|)
SELECT AVG(ABS(a.prediction - B.target)) mae
  FROM test_reg_score A natural join test_data B;

MAE             |   Umrechnung in ANZ_MOVIE
--------        |   ----------------------
.07584          |   778
```

Listing 4.11: Genauigkeit des Regressionsmodells

Neben der reinen Berechnung der Genauigkeit der Regressionsfunktion lassen sich die realen Werte und die vorhergesagten Werte ausgeben. In Listing 4.12 sind die Testdaten und deren Abweichung in Prozent (%) unter Anwendung der linearen Regressionsfunktion (Abb. 4.3) dargestellt.

Ist die Genauigkeit des Modells ausreichend, so kann diese für weitere Vorhersagen benutzt werden, d. h. man gibt dem Modell eine Menge von Jahreszahlen und lässt sich die voraussichtliche Anzahl an Filmen pro Jahr berechnen. Für die Vorhersage wird ebenfalls die Methode `DBMS_DATA_MINING.APPLY` (siehe Listing 4.11) verwendet.

```
YEAR        Vorhersage   Realer Wert     DIFF   ABWEICHUNG
---------  -----------   -----------    ------ ----------
 1991          6443           6503         60      0.92%
 1993          7742           7001        741     10.58%
 1995          9041           8210        831     10.12%
 1999         11639          11949        310      2.59%
 2004         14886          16462       1576      9.57%
 2005         15535          16688       1153      6.91%
```

Listing 4.12: Vergleich: reale Werte vs. vorhergesagte Werte (Testdaten) mit linearer Funktion

5. Darstellung & Auswertung der Ergebnisse: Die Regressionsfunktion kann nur vom Benutzer eingesehen werden, wenn bei der Konstruktion der lineare Kernel verwendet worden ist. Die Angabe der Regressionsfunktion beschränkt sich auf die Auflistung der Koeffizienten für die einzelnen Attribute. In Listing 4.13 wird die SQL--Anfrage gezeigt, die die Informationen also die Koeffizienten für die Regressionsfunktion liefert. Die Zusammensetzung (Interpretation) der Koeffizienten ergibt in diesem Beispiel folgende Regressionsfunktion f:

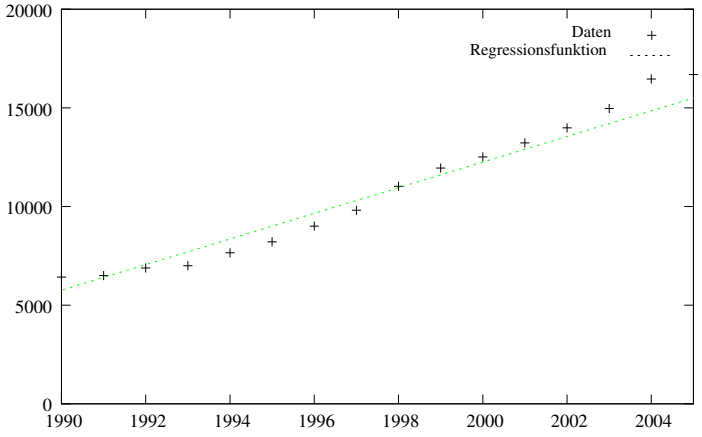


Abbildung 4.3: Regressionsfunktion

$$f(AnzahlderFilme) = 649.4 \times YEAR - 1286548$$

In Abb. 4.3 ist diese Regressionsfunktion graphisch abgebildet. Beim Gauss-Kernel arbeitet die Regressionsfunktion nach dem Black-Box-Prinzip und es besteht keine Möglichkeit sich die Funktion anzusehen.

```
SELECT B.attribute_name, round(B.coefficient, 10) as coefficient
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_SVM('test_reg_1')) A,
     TABLE(A.attribute_set) B;

-- normierte Werte (links) und reale Werte nach Umrechnung (rechts)
ATTRIBUTE_NAME    COEFFICIENT   | ATTRIBUTE_NAME    COEFFICIENT
----------------  -----------   | ----------------  -----------
YEAR               .9488868     | YEAR                649.4
                  -.0611781     |                    -1286548
```

Listing 4.13: Definition der Regressionsfunktion

4.2.2 Erfahrungen mit der Regressionsanalyse in ODM

Bei der Regressionsanalyse in ODM, die den Support Vector Machine-Algorithmus verwendet, wird bei vielen Problemen bevorzugt der Gauss-Kernel benutzt, da dieser flexibler als der lineare Kernel ist. Beim Gauss-Kernel besteht jedoch nicht die Möglichkeit die konstruierte Regressionsfunktion zu betrachten und damit die Rechnung nachzuvollziehen. Die eigentliche Abhängigkeit einer Variablen bzw. eines Attributs von anderen Attributen kann zwar berechnet werden, aber man erfährt nicht wie. Der lineare Kernel hingegen ermöglicht es, durch Betrachtung der Koeffizienten für die jeweiligen Attribute, die Regressionsfunktion zu verstehen und die Berechnung nachzuvollziehen. Jedoch lassen sich reale Probleme kaum durch lineare Funktionen beschreiben (siehe Kapitel 2.6.4). Die Verwendung kann aber durch eine explizite Angabe in der Konfigurationstabelle (siehe oben) erzwungen werden.

4.3 Clustering

In der Movie-Datenbank soll eine demographische Analyse der Schauspieler vorgenommen werden, bei der die Personen (eingeschränkt auf Schauspieler) zu Cluster zusammengefasst werden sollen. Das Ziel des Clustering besteht in einer Unterteilung der Schauspieler in unterschiedliche Gruppen und darüber hinaus um eine demographische Beschreibung dieser Gruppen.

Das Vorgehen des Clustering und letztlich die demographische Einteilung der Schauspieler wird in die folgenden Schritte unterteilt:

1. *Bereitstellung der Daten*
2. *Preprocessing und Transformation*
3. *Data Mining – Clustering*
4. *Darstellung & Auswertung der Ergebnisse*

4.3.1 Clustering: `Demographische Analyse der Schauspieler`

1. Bereitstellung der Daten: Im ersten Schritt werden alle Daten bezüglich der Aufgabe gesammelt und in einer Tabelle zusammengefasst. Für die demographische Einteilung der Schauspieler werden dazu folgende personenbezogene Attribute (Informationen) betrachtet. Dabei werden lediglich solche Personen in die demographische Analyse aufgenommen, deren personenbezogene Informationen vollständig gegeben sind, d. h. die Daten dürfen keine `NULL`-Werte besitzen.

1. Größe (Height)
2. Geschlecht (Sex)

1. Ist_Pseudonym_vorhanden (pseudo)
2. Alter (Age)

In Listing 4.14 ist die Datenvorbereitung dargestellt. Die Auswahl der Daten ist zusätzlich eingeschränkt auf Schauspieler, die nach 1950 geboren sind.

```
CREATE TABLE clus_data_sample_8 AS
  select a.person as id, sex, height,
         nvl2(realname, '1', '0') as pseudo,
         (to_number(to_char(sysdate, 'YYYY')) -
          to_number(to_char(birth_date, 'YYYY'))) ) as age
   from moviedb.person a
   where height is not null and sex is not null
    and birth_date is not null and mentioned = 'actor'
    and to_number(to_char(birth_date, 'YYYY')) > 1950
```

Listing 4.14: Datenvorbereitung für das Clustering

2. Preprocessing und Transformation: In diesem Schritt werden die Daten für das Clustering vorbereitet. Beim Preprocessing werden Ausreißer behandelt, wodurch extreme Werte speziell bei den Attributen Height und Age eingegrenzt werden sollen. Eine Behandlung von fehlenden Werten ist nicht notwendig, da die Daten durch die Zusammenstellung keine fehlenden Werte besitzen. Listing 4.15 zeigt die Behandlung der Ausreißer im Preprocessing-Schritt, wobei 2.5% der Daten als Ausreißer betrachtet werden sollen.

```
BEGIN
  DBMS_DATA_MINING_TRANSFORM.CREATE_CLIP('conf_clus_sample_8_1');
  DBMS_DATA_MINING_TRANSFORM.INSERT_CLIP_WINSOR_TAIL(
       'conf_clus_sample_8_1',      -- Konfiguration
       'clus_data_sample_8',
       0.025,                       -- prozentualer Anteil
       DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST(
                 'id',              -- Ausnahmen
                 'pseudo')
  );
  DBMS_DATA_MINING_TRANSFORM.XFORM_CLIP(
       'conf_clus_sample_8_1',
       'clus_data_sample_8',
       'clus_temp_1'                -- Ergebnis
  );
END;
```

Listing 4.15: Preprocessing - Behandlung der Ausreißer

Anschließend erfolgt die Transformation der Daten, die notwendig ist, da sowohl numerische als auch nicht-numerische Attribute enthalten sind. Da nicht-numerische Attribute intern mit 0 (nicht vorhanden) und 1 (vorhanden) kodiert werden, müssen die numerischen Attribute ebenfalls diesen Wertebereich haben. Um dies zu erreichen, werden die numerischen Attribute mit der Min-Max-Normierung auf den Bereich $[0, 1]$ transformiert. Dadurch haben alle Attribute den gleichen Einfluss auf die Abstandsfunktion (bei der Verwendung von Enhanced k-Means Algorithmus). In Listing 4.16 wird die Transformation der Daten dargestellt.

```
BEGIN
  DBMS_DATA_MINING_TRANSFORM.CREATE_NORM_LIN('conf_clus_sample_8_2');
  DBMS_DATA_MINING_TRANSFORM.INSERT_NORM_LIN_MINMAX(
       'conf_clus_sample_8_2',
       'clus_temp_1',
       DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST(
                        'id')
  );
  DBMS_DATA_MINING_TRANSFORM.XFORM_NORM_LIN(
       'conf_clus_sample_8_2',
       'clus_temp_1',
       'clus_prepared'
  );
END;
```

Listing 4.16: Transformation - Normierung der numerische Werte

3. Data Mining – Clustering: Nachdem die Daten vorbereitet sind, kann die demographische Einteilung der Schauspieler durchgeführt werden. Für das Beispiel soll das Clustering mit dem k-Means-Algorithmus durchgeführt werden. Die Einteilung der Schauspieler soll in acht Gruppen erfolgen und diese Einstellung wird entsprechend in der Konfigurationstabelle vorgenommen, indem der Parameter `clus_num_clusters` gleich acht gesetzt wird. Außerdem wird die Anzahl der durchzuführenden Iterationen zur Findung der Cluster, dessen Parameter `kmns_iterations` einen Wert zwischen 0 und 20 annehmen kann, auf fünf gesetzt. Ansonsten bleiben die voreingestellten Werte für die Parameter unverändert. Listing 4.17 stellt die notwendigen Schritte für das Clustering dar.

```
BEGIN
    INSERT INTO test_clus_settings
       VALUES (DBMS_DATA_MINING.algo_name,
                DBMS_DATA_MINING.algo_kmeans);
    INSERT INTO test_clus_settings
       VALUES(DBMS_DATA_MINING.clus_num_clusters,8);
    INSERT INTO test_clus_settings
       VALUES (DBMS_DATA_MINING.kmns_iterations,5);

    DBMS_DATA_MINING.CREATE_MODEL(
       model_name               => 'clus_model',
       mining_function          => DBMS_DATA_MINING.CLUSTERING,
       data_table_name          => 'clus_prepared',
       case_id_column_name      => 'id',
       settings_table_name      => 'test_clus_settings'
    );
END;
```

Listing 4.17: Clustering - K-Means mit $k = 8$

4. Darstellung & Auswertung der Ergebnisse: Die Darstellung der Ergebnisse des Clustering kann durch die Ausgabe der Cluster-Hierarchie (siehe Kapitel 3) und der Cluster-Regeln erfolgen. Aus der Cluster-Hierarchie, die alle erstellten Cluster enthält, werden im folgenden Listing 4.18 nur die gesuchten Cluster betrachtet (demographische Einteilung in 8 Gruppen). Dabei liefert die Anfrage

in Listing 4.18 die Cluster-Kennung (RULE_ID), sowie den Support (=Anzahl der Daten im jeweiligen Cluster) und die Confidence (= Aussagekraft der Cluster-Regel zur Beschreibung des Clusters, z. B. 90% der Daten im Cluster *X* entsprechen der Beschreibung) der Cluster.

```
-- Auflistung der Cluster-Regeln und deren Aussagekraft
SELECT A.rule.rule_id,          -- Identifikation
       A.rule.rule_support,     -- Unterstützung durch Daten (Anzahl)
       A.rule.rule_confidence   -- Glaubwürdigkeit der Regel
FROM table(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM ('clus_model')) A,
      table(A.child) B
WHERE B.ID IS NULL;

RULE_ID    RULE_SUPPORT    RULE_CONFIDENCE
-------    ------------    ---------------
  8            2886            .9694
  9            3051            .9010
 10            3767            .9548
 11            4481            .9036
 12            1399            .9517
 13            1935            .9723
 14            2078            .9034
 15            3169            .9077
```

Listing 4.18: Clustering - demographische Einteilung der Schauspieler

Die Cluster definieren in diesem Beispiel die demographische Einteilung der Schauspieler in repräsentative Gruppen und werden durch die Cluster-Regeln definiert. Listing 4.19 veranschaulicht beispielhaft die Ausgabe einer Cluster-Regel für den Cluster mit der ID=10.

Dieser Cluster fasst alle Schauspieler zu einer Gruppe zusammen, die männlich sind, kein Pseudonym benutzen, deren Größe zwischen (0.4) und (1) und deren Alter zwischen (0.5) und (1) liegt. Dabei ist zu beachten, dass die Werte für Größe und Alter normiert sind und entsprechend zurückgerechnet werden müssen.

```
SELECT  A.rule.rule_id as id, A.rule.rule_confidence*100 || '%',
        B.attribute_name || B.conditional_operator ||
        nvl(B.attribute_str_value, B.attribute_num_value) as ANTE,
        C.attribute_name || C.conditional_operator ||
        nvl(C.attribute_str_value, C.attribute_num_value) as CONS
FROM table(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM ('clus_model'))A,
      table(A.rule.antecedent) B,
      table(A.rule.consequent) C,
      table(A.child) D
WHERE D.ID IS NULL;

ID   CONFIDENCE        ANTE               CONS
--   ----------        ---------------    ---------------
10     95.48%          AGE >=  .5         Cluster = 10
10     95.48%          AGE <= 1           Cluster = 10
10     95.48%          HEIGHT  >= .4      Cluster = 10
10     95.48%          HEIGHT  <=  1      Cluster = 10
```

```
10      95.48%          PSEUDO IN  0          Cluster = 10
10      95.48%          SEX IN  m             Cluster = 10
```

Listing 4.19: Clustering - Ergebnis-Cluster

Anmerkung: Ein weiteres Vorgehen beim Clustering könnte beispielsweise in die Hinzunahme bzw. Wegnahme von Attributen erfolgen, wodurch sich Cluster mit veränderten Beschreibungen ergeben. Unabhängig vom Ergebnis müssen die Cluster jeweils interpretiert werden, um diese weiterverwenden zu können.

4.3.2 Erfahrungen mit der Clusteranalyse in ODM

- Beim Versuch das Clustering mit dem O-Cluster-Algorithmus durchzuführen, endete dieser stets mit dem Abbruch des Clustering mit der folgenden Fehlermeldung in Listing 4.20. Der Algorithmus scheint intern ein Problem mit der Diskretisierung (Zeile 4 – `invalid bin number`) zu haben und bricht aus diesem Grund ab.[1]

```
1 ORA-06512: at "SYS.DBMS_SYS_ERROR", line 105
2 ORA-06512: at "DMSYS.ODM_OC_CLUSTERING_MODEL", line 122
3 ORA-06512: at "DMSYS.ODM_OC_CLUSTERING_MODEL", line 2312
4 ORA-40321: invalid bin number, is zero or negative value
5 ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
6 ORA-06512: at "DMSYS.ODM_OC_CLUSTERING_MODEL", line 630
7 ORA-06532: Subscript outside of limit
8 ORA-06512: at "DMSYS.DBMS_DATA_MINING", line 305
```

Listing 4.20: Clustering - O-Cluster-Algorithmus

- Beim Versuch eine Menge mit dem Enhanced k-Means-Algorithmus in 8 Cluster zu unterteilen, erzeugte der Algorithmus die gewünschte Anzahl an Cluster, jedoch wurde die gesamte Menge genau einem Cluster zugeordnet. Dieses Vorgehen verstößt aber gegen die Definition eines Clusters (siehe Kapitel 2), wonach jeder Cluster mind. ein Element enthalten muss. In Listing 4.21 ist die dazugehörige Cluster-Hierarchie abgebildet, wobei die Cluster mit ID = {2,4,6,8,10,12,14,15}, die gesuchten Cluster sind. Eine Anfrage zur Auflistung der Cluster-Regeln (Zeile 15 ff) liefert dabei außerdem lediglich die Regel für den Cluster mit der ID=15.

```
1  SELECT id, parent, tree_level as level, record_count as count
2  FROM table(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM('model'));
3
4  ID  PARENT LEVEL COUNT   |   ID  PARENT LEVEL COUNT
5  --  ------ ----- -----   |   --  ------ ----- -----
6  1           1   7235    |    9    7      5   7235
7  2     1     2   0       |   10    9      6   0
8  3     1     2   7235    |   11    9      6   7235
9  4     3     3   0       |   12   11      7   0
10 5     3     3   7235    |   13   11      7   7235
11 6     5     4   0       |   14   13      8   0
12 7     5     4   7235    |   15   13      8   7235
13 8     7     5   0       |
```

[1] Das Beispiel aus der Oracle Data Mining-Cartridge funktioniert jedoch aus unersichtlichen Gründen

```
14 -- Ausgabe der Cluster-Regeln für die 10 Cluster
15 SELECT A.rule.rule_id as id, A.rule.rule_support as support,
16        A.rule.rule_confidence*100 || '%' as confidence
17 FROM table(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM ('model')) A,
18       table(A.child) B
19 WHERE B.ID IS NULL;
20
21  ID         SUPPORT           CONFIDENCE
22 ----        -------           ----------
23  15           6766                93%
```

Listing 4.21: Clustering - K-Means-Algorithmus

4.4 Klassifikation

Eine weitere interessante Fragestellung ergibt sich, wenn der Einfluss von Schauspielern, die in einem Film mitgespielt haben, auf den Erfolg eines Filmes untersucht wird. Diese Frage lässt möglicherweise sich mit der Klassifikation als Data Mining-Methode beantworten. Die Daten, die für die Klassifikation notwendig sind, sind die Schauspieler und das Rating (Bewertung) bezogen auf den entsprechenden Film.

Die Klassifikation erfolgt wiederum in mehreren Schritten, die im folgenden aufgelistet und beschrieben sind:

1. *Bereitstellung der Daten*
2. *Preprocessing und Transformation*
3. *Data Mining – Klassifikation*
4. *Verifizierung des Klassifikators (Genauigkeit)*
5. *Klassifikation neuer Filme*

4.4.1 Klassifikation: `Einfluss von Schauspielern auf die Bewertung des Films`

1. Bereitstellung der Daten: Im ersten Schritt werden wieder alle notwendigen Information für die Data Mining-Aufgabe gesammelt und in einer Tabelle zusammengefasst. Für die Klassifikation sind die Schauspieler, die in einem Film mitgespielt haben, und die Bewertung des jeweiligen Films zusammenzuführen. Die Datenmenge die letztlich zur Konstruktion des Klassifikators verwendet werden soll, lässt sich mit den folgenden Punkten beschreiben:

1. Ein Schauspieler muss in mind. 10 Filmen eine gekennzeichnete Position im Abspann haben.
2. Der Film muss mind. 1000 Bewertungen haben.
3. Der Film muss ein Kinofilm sein, also vom Typ `cinema`.

2. Preprocessing und Transformation: Ein Preprocessing ist in diesem Fall nicht durchzuführen, da durch die Zusammenstellung der Daten keine fehlende Werte enthalten sein können. Die Behandlung von Ausreißern wird ebenfalls nicht durchgeführt, da in diesem Beispiel für die Klassifikation keine numerischen Werte enthalten sind.

Für die vorgesehene Klassifikation und in Anbetracht dessen, dass das Klassifikationsattribut numerisch ist, erscheint es jedoch notwendig eine Transformation in Form einer Diskretisierung des Klassifikationsattributs `Rating` (Bewertung) durchzuführen. Die Diskretisierung wird dabei in die

Klassen `niedrig`, `mittel` und `hoch` vorgenommen, wobei die folgenden Wertebereiche gelten sollen: `niedrig` $\in [1,4]$, `mittel` $\in (4,7]$ und `hoch` $\in (7,10]$.

Bei der Fragestellung nach dem Einfluss der Schauspieler auf die Bewertung eines Films geht es nicht nur um eine direkte Beziehung, d. h. Schauspieler `X` spielt in Film `Y` mit der Bewertung `Z`, sondern auch um eine Kombination von Schauspieler, die in einem Film mitgespielt haben, d. h. Schauspieler `U,V,W` und `X` haben in Film `Y` mit der Bewertung `Z` mitgespielt.

Die Daten müssen also wieder derart transformiert werden, dass sie dem Charakter eines Warenkorbs mit Waren (siehe Kapitel 2 – Assoziationsanalyse) entsprechen, bei dem der Film den Warenkorb und die Schauspieler die Waren repräsentieren. Das Vorgehen dieser Transformation wurde bereits in Abschnitt 4.1.1 demonstriert. In Listing 4.22 wird lediglich die Transformation der Daten dargestellt.

```
CREATE TABLE class_prepared
 NESTED TABLE actor_for_movie STORE AS DM_Nested_Numericals_actor
 AS (
     SELECT id,
        CAST(COLLECT(DM_Nested_Numerical(
          SUBSTRB(name, 1, 30), 1))
        AS DM_Nested_Numericals) actor_for_movie,
        target
     FROM daten_tabelle
     GROUP BY id, target  );
```

Listing 4.22: Klassifikation - Transformation der Daten

3. Data Mining – Klassifikation: In diesem Schritt erfolgt die Konstruktion des Klassifikators auf Grundlage der vorbereiteten Daten. Die Daten werden dabei in Trainingsdaten und und Testdaten aufgeteilt, wobei sich der Anteil der Testdaten auf 4% der gesamten Daten beläuft.

Für die Konstruktion des Klassifikator kommt aufgrund der Verwendung einer geschachtelten Tabelle nur der Naive-Bayes-Algorithmus oder der Support-Vector-Machine-Algorithmus in Frage. Die Wahl fällt auf den Support-Vector-Machine-Algorithmus, da dieser im Gegensatz zum Naive--Bayes die notwendigen Parameter automatisch bestimmt und somit ein optimales Ergebnis – im Sinne der zur Verfügung stehenden Daten – liefert[BM05]. Listing 4.23 beschreibt die Konstruktion des Klassifikators.

```
BEGIN
  INSERT INTO class_settings (setting_name, setting_value)
      VALUES (dbms_data_mining.algo_name,
              dbms_data_mining.algo_support_vector_machines);

  DBMS_DATA_MINING.CREATE_MODEL(
      model_name              => 'classificator',
      mining_function         => DBMS_DATA_MINING.CLASSIFICATION,
      data_table_name         => 'data_table_training',
      case_id_column_name     => 'id',     -- ID für Movie
      target_column_name      => 'target', -- Rating
      settings_table_name     => 'class_settings');
END;
```

Listing 4.23: Konstruktion des Klassifikators

4. Verifizierung des Klassifikators: Nach der Konstruktion des Klassifikator soll anschließend die Zuverlässigkeit (Genauigkeit) gemessen werden. Entspricht die Genauigkeit nicht den gesetzten Erwartungen, so müssen ggf. Modifikationen in den vorhergehenden Schritten (siehe Kapitel 2) vorgenommen werden.

Bei der Berechnung der Genauigkeit des Klassifikators werden zwei Unterscheidungen gemacht. Im ersten Fall wird die Genauigkeit für die gesamten Daten berechnet, d. h. die Trainingsdaten und die Testdaten zusammen. Und im zweiten Fall wird die Genauigkeit des Klassifikators, eingeschränkt auf die Testdaten, berechnet.

Durch den Vergleich dieser Werte kann man erkennen, ob der Klassifikator möglicherweise an Overfitting 2.6.4 leidet. In Listing 4.24 sind die Anfragen zur Berechnung der Confusion-Matrix und die Genauigkeit des Klassifikators dargestellt. Aus der Betrachtung der Werte für die Genauigkeit ist zu erkennen, dass der Wert für die Trainingsdaten und Testdaten zusammen wesentlich höher liegt als nur für die Testdaten. Daraus ist erkennbar, dass der Klassifikator schlecht mit unbekannten Daten umgehen kann und somit unter Overfitting leidet.

```
-- Berechnung der Confusion-Matrix
SELECT target,(PREDICTION(classificator USING *) AS prediction,
      COUNT(*) AS Anzahl
FROM data_table    (respektive data_table_test)
GROUP BY target, PREDICTION(classificator USING *);

-- Bei Anzahl sind die Werte für Complete (links) und Test (rechts)
TARGET    PREDICTION   ANZAHL  | TARGET    PREDICTION  ANZAHL
--------  ----------   ------- | --------  ----------  ------
niedrig   niedrig         207  | niedrig   niedrig        0
niedrig   mittel           33  | niedrig   mittel         1
niedrig   hoch              7  | niedrig   hoch           1
mittel    niedrig           1  | mittel    niedrig        1
mittel    mittel         3251  | mittel    mittel       115
mittel    hoch             48  | mittel    hoch          30
hoch      niedrig           1  | hoch      niedrig        1
hoch      mittel           78  | hoch      mittel        27
hoch      hoch           1957  | hoch      hoch          48

-- Berechnung der Genauigkeit
SELECT SUM(DECODE(target,
          PREDICTION(classificator USING *), 1, 0))
     /count(*) *100 || '%' as Accuracy
FROM data_table;   (respektive data_table_test)

ACCURACY (Complete)  |  ACCURACY (Test)
-------------------  |  ----------------
97%                  |   72%
```

Listing 4.24: Klassifikation - Transformation der Daten

5. Klassifikation neuer Filme: Nachdem der Klassifikator definiert ist und die Genauigkeit für die gestellte Aufgabe akzeptabel ist, können in diesem Schritt neue noch nicht klassifizierte Daten klassifiziert werden. Dabei ist jedoch aufgrund des konstruierten Klassifikators zu beachten, dass nur

dann eine glaubwürdige Klassifikation möglich ist, wenn die neuen Filme, deren Bewertung vorhergesagt werden soll, sich aus den Schauspielern zusammensetzt, die dem Klassifikator bekannt sind.

Die Klassifikation neuer Daten ist in Listing 4.25 beispielhaft dargestellt. Das Ergebnis wird in einer Tabelle abgespeichert und kann von dort weiterverarbeitet werden, z. B. kann ein neuer aktueller Kinofilm, deren Schauspieler (auch eine Teilmenge der Schauspieler ist denkbar) bekannt sind, benutzt werden, um eine Bewertung zu bekommen und danach zu entscheiden, ob man sich den Film ansieht oder nicht.

Als Beispiele für die Anwendung des Klassifikators sollen die Filme *Night at the Museum (2006)* und *El Laberinto del Fauno (2006)* dienen. Zur Beurteilung der Filme werden die Hauptdarsteller der Filme verwendet, die in den folgenden Auflistungen zu finden sind.

a) Night at the Museum (2006) (mit ID=1001)

- Ben Stiller
- Dick Van Dyke
- Mickey Rooney
- Bill Cobbs
- Ricky Gervais

b) El Laberinto del Fauno (2006) (mit ID=1002)

- Ariadna Gil
- Sergi López
- Maribel Verdú
- Doug Jones
- Álex Angulo

```
-- Vorbereiten der Filme und der Schauspieler
-- für die anschließende Klassifikation
BEGIN
    DBMS_DATA_MINING.APPLY(
       model_name          => 'classificator',
       data_table_name     => 'data_table_new',
       case_id_column_name => 'id',
       result_table_name   => 'movie_apply_result'
    );
END;

-- Ausgabe der wahrscheinlichsten Klassifikation der Filme
SELECT id, prediction, probability*100 || '%' as probability
FROM (
       SELECT id, max(probability) as probability
       FROM movie_apply_result
       GROUP BY id ) NATURAL JOIN movie_apply_result;

ID         PREDICTION      PROBABILITY    (IMDB)
-------    ----------      -----------    ------------
1001         mittel          62%           6.4 (mittel)
1002         hoch            58%           8.5 (hoch)
```

Listing 4.25: Klassifikation unbekannter Daten

Eine andere Anwendung des Klassifikators wäre z. B. die mögliche Klassifikation eines Films, indem Schauspieler im Vorhinein ausgewählt werden, um dadurch die Chancen auf einen erfolgreichen Film auszuloten. In Listing 4.26 ist ein Ergebnis für eine derartige Anfrage dargestellt, die untersucht, ob ein Film mit den Schauspielern 1. *Matt Damon* und 2. *Humphrey Bogart* Erfolg haben würde (*Anmerkung:* Sehr hypothetisch!).

```
-- Anwendung des Klassifikators mittels der APPLY-Funktion
-- ID ist ein künstliches Attribut zur Identifikation
SELECT id, prediction, probability*100 || '%' as probability
FROM movie_apply_result;

  ID            PREDICTION            PROBABILITY
  -------       -------------         -------------
  1003           hoch                    50%
  1003           mittel                  35%
  1003           niedrig                 15%
```

Listing 4.26: Bewertung eines Films mit einer fiktiven Kombination von Schauspielern

4.4.2 Erfahrungen mit der Klassifikation in ODM

- Die Klassifikation in Oracle Data Mining bietet durch die vier implementierten Algorithmen eine große Auswahl und macht somit eine breite Anwendung möglich. Jedoch kann es problematisch werden, wenn man eine der verfügbaren Methoden favorisiert, da beispielsweise die Verarbeitung von geschachtelten Tabellen lediglich auf den Support-Vector-Machine-Algorithmus und den Naive Bayes-Algorithmus beschränkt ist. Und die Transparenz der Methode ist nur beim Klassifikationsbaum, im Single-Feature-Modus vom Adaptive Bayes Network und beim linearen Kernel der Support-Vector-Machines gegeben.
- Die Konstruktion des Klassifikators und die anschließende Klassifikation ist sehr schnell, so dass die durchgeführten Beispiele weniger als 20sec für die Konstruktion respektive Klassifikation benötigten.
- Bei der Klassifikation mit dem Klassifikationsbaum kann der Klassifikationsprozess u. a. durch die Klassifikationsregeln nachvollzogen werden. Listing 4.27 enthält dazu beispielhaft eine Klassifikationsregel, die in der Form von Oracle Data Mining durch das *Parsen* des XML-Baums (siehe Kapitel 3) erzeugt worden ist. Bei dieser Regel ist bereits zu erkennen, dass diese noch vereinfacht werden könnte. Oracle Data Mining fasst also einfach alle Entscheidungen vom Wurzelknoten bis zum Blattknoten zu einer Klassifikationsregel zusammen. Ein Post-Processing zur Vereinfachung der Regeln findet nicht statt, wodurch die Lesbarkeit und das Verständnis gemindert wird.

```
-- Klassifikationsregel 1 (Oracle)
IF ATTRIBUTE_1 <= VALUE_1 AND ATTRIBUTE_1 <= VALUE_2 AND
   ATTRIBUTE_2 <= VALUE_3 AND ATTRIBUTE_2 >  VALUE_4 AND
   ATTRIBUTE_3 is in {VALUE_5 VALUE_6} AND
   ATTRIBUTE_3 is in {VALUE_5 VALUE_6 VALUE_7 VALUE_8}
THEN
     CLASS_A
END;
-- Vereinfachung der Klassifikationsregel 1
IF ATTRIBUTE_1 <= VALUE_1 AND ATTRIBUTE_2 <= VALUE_3 AND
   ATTRIBUTE_2 > VALUE_4 AND ATTRIBUTE_3 is in {VALUE_5 VALUE_6}
THEN
     CLASS_A
END;
```

Listing 4.27: Problem bei Klassifikationsregeln in Oracle Data Mining

5 Data Mining in räumlichen Datenbanken

Der größte Anteil, etwa 80% ([SC03]), der heute digital gespeicherten Informationen sind in Datenbanken gespeichert, die einen räumlichen Anteil haben, und werden entsprechend räumliche Datenbanken (*Spatial Databases*) genannt. Dabei wird der Begriff des Raumes in diesem Zusammenhang stets mit der geographischen Interpretation des Raumes, d. h. $\mathbb{R}^2$ oder $\mathbb{R}^3$, verwendet. Anwendungsgebiete von räumlichen Datenbanken sind die Klimaforschung, Wetterbeobachtung, Geo-Marketing, Verkehrskontrolle u. v. m. Die Grundlage für räumliche Daten liefern beispielsweise die Earth Observation Satellites (EOS), welche die Oberfläche der Erde systematisch erfassen, was pro Tag ungefähr ein Terabyte an Informationen ausmachen kann. Aufgrund dieser riesigen Datenmenge ist zu erwarten, dass darin eine Vielzahl an interessanten, jedoch unbekannten, Informationen enthalten ist. Die Knowledge Discovery, wie sie in Kapitel 2 beschrieben wurde, kann in dieser Form jedoch nicht ohne weiteres auf räumliche Datenbanken angewendet werden. Aus diesem Grund müssen die Techniken und Methoden der Knowledge Discovery an die räumlichen Datenbanken angepasst und erweitert werden.

In diesem Kapitel liegt demnach das Hauptaugenmerk auf der Knowledge Discovery und dem Data Mining in räumlichen Datenbanken, sowie den damit verbundenen Herausforderungen. Ferner soll der Unterschied zwischen dem Data Mining in relationalen Datenbanken (im Weiteren als klassisches Data Mining bezeichnet) und dem Data Mining in räumlichen Datenbanken (Spatial Data Mining) verdeutlicht werden.

Die weiteren Punkte des Kapitels gliedern sich folgendermaßen: Zunächst werden die Begriffe räumliche Datenbanken und Spatial Data Mining kurz erläutert, woraufhin anschließend unterschiedliche Lösungsvorschläge aus der Literatur für das Data Mining in räumlichen Datenbanken aufgeführt und beschrieben werden. Aufbauend auf den Lösungsvorschlägen aus der Literatur wird ein Konzept zur Materialisierung (Transformation) von räumlichen Daten und Informationen entwickelt, so dass anschließend klassisches Data Mining, mit den in Kapitel 2 vorgestellten Methoden, durchgeführt werden kann.

5.1 Räumliche Datenbanken

Räumliche Datenbanken (*spatial databases* besitzen im Gegensatz zu *normalen* relationalen Datenbanken Attribute, die sich in nicht-räumliche (*non-spatial*) und räumliche (*spatial*) Attribute aufteilen. Die nicht-räumlichen Attribute charakterisieren Eigenschaften, welche nicht raumbezogen sind z. B. Name, Bevölkerung und Kriminalitätsrate einer Stadt, und nehmen einfache Datentypen, wie z. B. Zahlen (NUMBER, etc.) und Zeichenketten (VARCHAR, etc.) an. Im Gegensatz dazu charakterisieren die räumlichen Attribute die räumliche Lage (Längen- und Breitengrad) und geometrischen Eigenschaften, z. B. Höhenangabe und Geometrie (Form) und nehmen komplexere Datentypen an, z. B. Polygon, Linie, Punkt.

Die Kombination der räumlichen und nicht-räumlichen Attribute wird in räumlichen Datenbanken als Schicht (*Layer*) bezeichnet. Dabei ist charakteristisch, dass eine Schicht immer genau ein räumliches Attribut und mehrere nicht-räumliche Attribute, die sich auf das räumliche Attribut beziehen, besitzt. Die Realisierung der Schichten in der Datenbank erfolgt durch Relationen (d. h. Tabellen).

Eine Schicht wird auch als *thematische Karte* bezeichnet. Für die Darstellung der thematischen Karten gibt es zwei Möglichkeiten – Raster-Darstellung und Vektor-Darstellung [KAH96]:

1. In der Raster-Darstellung werden die räumlichen Attribute durch Punkte dargestellt und die nicht-räumlichen Attribute werden durch Intensitäten oder Farben repräsentiert.
2. In der Vektor-Darstellung werden die räumlichen Attribute durch Geometrien dargestellt (i. d. R. durch die Begrenzung der Geometrie), die mit nicht-räumlichen Attributen annotiert sind.

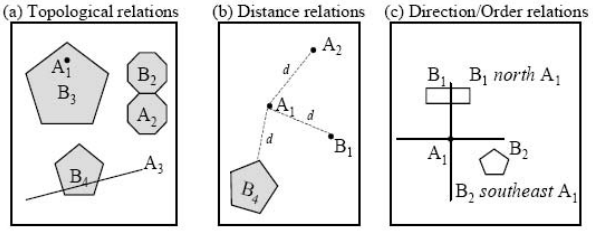


Abbildung 5.1: Räuml. Beziehungen[VB05a]

Die Daten in einer räumlichen Datenbank beinhalten darüber hinaus implizite Beziehungen, die die Lage und Nachbarschaft der Objekte zueinander beschreiben. Nach [EKS97] [KH95] [SC03] lassen sich die impliziten Beziehungen in die folgenden drei Gruppen (siehe Abb. 5.1) einteilen: a) topologische Beziehungen, z. B. A_2 *benachbart* B_2, B_3 *enthält* A_1 oder A_3 schneidet B_4, b) metrische Beziehungen, z. B. Abstand (d = Abstand von A_1 nach A_2) , Durchmesser, Fläche (Flächeinhalt B_4), und c) gerichtete Beziehungen, z. B. B_2 *SüdWest_von* A_1, B_2 *Nord_von* A_1 .

a.) **topologische Beziehungen** – Die topologischen Beziehungen zweier Objekte ergeben sich aus der Betrachtung der möglichen Überlagerungen der Objekte. Für zwei-dimensionale Objekte A und B ergeben sich nach [Ege91] acht topologische Beziehungen: A *disjunkt* B, A *berührt* B, A *überdeckt* B, A *wird_überdeckt_von* B, A *innerhalb* B, A *gleich* B, A *enthält* B und A *überlappt* B.

   Diese Beziehungen werden nach [Ege91] in einem Modell, dem sogenannten *9-Intersection-Modell* (9IM), zusammengefasst. Einige dieser Beziehungen sind jedoch redundant, z. B. A *überdeckt* B, A *wird_überdeckt_von* B, weshalb in [YLG+05] bewiesen wird, dass lediglich fünf topologische Beziehungen zwischen zwei zwei-dimensionalen Objekten nötig sind: A *disjunkt* B, A *berührt* B, A *gleich* B, A *innerhalb* B und A *überlappt* B.

b.) **metrische Beziehungen** – Die metrische Beziehung beschreibt geometrische Berechnungen, die durch die Koordinaten zweier Objekte und ein Referenz-System, z. B. das metrisches System, bestimmt werden. In Bereichen mit geographischem Bezug findet die euklidische Geometrie mit dem euklidischen Abstand (siehe Kapitel 2.6.2), zur Berechnung von metrischen Beziehungen, Anwendung.

c.) **gerichtete Beziehungen** – Die gerichtete Beziehung beschreibt die relative Lage von zwei Objekten zueinander im geometrischen Raum, z. B. A *NordWest_von* B. Zur exakten Beschreibung einer gerichteten Beziehung sind zwei Objekte und ein fester Referenzpunkt notwendig. Grundsätzlich lassen sich diese Beziehungen durch numerische Werte, z. B. Winkelangabe, oder durch Symbole (z. B. *Norden, Süden, Ost, West, rechts, links*) beschreiben.

5.2 Spatial Data Mining

Das Data Mining wird bei der Anwendung auf räumliche Datenbanken Spatial Data Mining genannt. Es drängt sich jedoch die Frage auf, warum das Data Mining in diesem Zusammenhang eine neue Bezeichnung erhält. Der Grund liegt zum einen in der Darstellung der räumlichen Daten, d. h. komplexe geometrische Daten, und zum anderen in der um räumliche Aspekte erweiterte Betrachtung, die in räumlichen Datenbanken implizit gespeichert sind.

In [HK00] wird das Spatial Data Mining folgendermaßen beschrieben: *Spatial Data Mining refers to the extraction of knowledge, spatial relations, or other interesting patterns not explicitly stored in spatial databases.* Im Folgenden sind die allgemeinen Punkte aufgeführt, die den entscheidenden Unterschied zwischen klassischem und Spatial Data Mining ausmachen:

- In räumlichen Datenbanken, die räumliche Zusammenhänge der Realität abbilden, ist die Eigenschaft enthalten, dass sich räumliche Objekte implizit gegenseitig beeinflussen. Dieser Aspekt ist bekannt als das Erste Gesetz der Geographie: *Everything is related to everything else, but nearby things are more related than distant things.*[Tob79] und wird auch räumliche Autokorrelation genannt.
- Die Annahme, dass alle Daten unabhängig voneinander erzeugt worden sind, gilt lediglich beim klassischen Data Mining in relationalen Datenbanken. In räumlichen Datenbanken ist die statistische Unabhängigkeit der Daten nicht gegeben [SC03].

Durch die Betrachtung räumlicher Aspekte (d. h. ein Ort und seine Beziehungen zu anderen Orten) lassen sich viele soziale, ökonomische und ökologische Phänomene in unserer Umwelt besser verstehen und erklären. Die genannten Merkmale von raumbezogenen Objekten sind entscheidend für das Spatial Data Mining und müssen berücksichtigt werden. Beispielsweise könnte das implizite Wissen ohne diese räumliche Betrachtung in den folgenden Beispielen nicht entdeckt werden.

Beispiele: [SC03]

(a) Im Jahr 1855 grassierte die asiatische Cholera in London. Ein Epidemiologe markierte daraufhin alle Orte auf einer Karte, in denen ein Krankheitsfall gemeldet wurde. Die Markierungen auf der Karte formten sich zu Cluster, in deren Zentren sich jeweils ein Brunnen befand, welche Auslöser der Epidemie waren.

Entscheidend hierbei ist der räumliche Zusammenhang, der sich aus der räumlichen Verteilung der Krankheitsfälle um die städtischen Brunnen für die Wasserversorgung ergibt.

(b) Im Jahr 1909 machte eine Gruppe von Zahnärzten eine interessante Entdeckung. Sie fanden heraus, dass die Menschen in Colorado Springs ungewöhnlich gesunde Zähne hatten. Eine spätere Studie ergab, dass das Trinkwasser einen besonders hohen Anteil an Fluoriden aufwies.

In diesem Fall ist ebenfalls entscheidend, dass sich der räumliche Zusammenhang daraus ergibt, dass die Personen mit den ungewöhnlich gesunden Zähnen in der räumlichen Umgebung eines außerordentlichen Trinkwasser-Speichers anzutreffen waren.

Diese Beispiele zeigen bereits, dass mit dem Spatial Data Mining ein besseres Verständnis von räumlichen und nicht-räumlichen Daten, ausgedrückt in räumlichen Zusammenhängen und Beziehungen, erreicht werden kann. Außerdem stehen beim Spatial Data Mining primär Erkenntnisse und Zusammenhänge im Vordergrund, bei denen nicht-räumliche und räumliche Daten involviert sind.

Im Vergleich stellt sich das Data Mining in räumlichen Datenbanken deutlich komplexer und schwieriger als das klassische Data Mining dar. Folgende Punkte sollen Hinweise darauf geben, warum es so schwierig ist:

- komplexe Datenstrukturen, z. B. Objekte werden durch Polygone dargestellt
- rechenintensive Operatoren und Funktionen für geometrische Objekte
- Existenz von impliziten Beziehungen zwischen geometrischen Objekten
- Autokorrelation, d. h. die gegenseitige Beeinflussung räumlich benachbarter Objekte

- *Feature Selection*, d, h. die Auswahl der Daten und Attribute für das Data Mining, gestaltet sich schwierig aufgrund der Eigenschaft der Autokorrelation

Anmerkung:

Um die Unterscheidung zwischen klassischem und Spatial Data Mining zu verdeutlichen, soll der *Punkt*, der das einfachste geometrische Objekt darstellt, betrachtet werden. Der Punkt bezeichnet in der Mathematik einen nicht-ausgedehnten Ort in einem beliebigen Raum. Für den Fall, dass der Raum auf die geographischen Dimensionen beschränkt ist, wird der Punkt im zwei-dimensionalen Raum ($\mathbb{R}^2$) durch die Koordination x, y, im drei-dimensionalen Raum ($\mathbb{R}^3$) zusätzlich durch z beschrieben. Der Punkt stellt für sich gesehen eine Einheit dar, die einen Ort im Raum beschreibt.
Das klassische Data Mining, welches keine komplexe Datentypen verarbeiten kann (siehe Kapitel 2), also auch keinen Punkt, müsste für die Repräsentation des Punktes seine Koordinaten benutzen, d. h. jede Koordinate des Punktes entspricht einem Attribut. Das klassische Data Mining, z. B. Clustering, würde auch fehlerfrei funktionieren, jedoch geht, durch die Repräsentation des Punktes durch seine Koordinaten, die Semantik verloren, die möglicherweise von Bedeutung ist, wenn das Ergebnis des Data Mining ausgewertet wird, d. h. es werden keine Punkte sondern lediglich *zusammenhanglose* Attribute verarbeitet.
Ebenso könnte ein Rechteck, repräsentiert durch die Punkte der linken unteren und rechten oberen Ecke beschrieben und gleichfalls durch Attribute ausgedrückt werden, jedoch geht in diesem Fall erst recht die Semantik verloren, da das klassische Data Mining keine Rechtecke, sondern Eckpunkte dargestellt durch Koordinaten verarbeitet.

Im Folgenden werden die Methoden, die beim Spatial Data Mining eingesetzt werden, angegeben und beschrieben. Zum einen gilt, dass die Methoden, die bereits beim klassischen Data Mining eingesetzt wurden, auch in räumlichen Datenbanken eingesetzt werden können und demzufolge ein entsprechendes Gegenstück besitzen. Darüber hinaus können jedoch noch weitere Methoden eingesetzt werden, da durch die räumlichen Eigenschaften – vor allem die Autokorrelation – Muster in den Daten enthalten sind, die im klassischen Fall nicht vorhanden sind [SSG01]. Die folgende Auflistung gibt einen Überblick über die Methoden, die zum Data Mining in räumlichen Datenbanken eingesetzt werden:

1. Spatial Clustering
2. Räumliche Klassifikation und Regression
3. Räumliche Assoziationsanalyse
4. Spezielle Spatial Data Mining-Methoden
   a) Co-Location Analyse
   b) Räumliche Trend Analyse

5.2.1 Problemstellungen des Spatial Data Mining

In diesem Abschnitt sollen die Problemstellungen des Spatial Data Mining herausgestellt werden. Dabei werden die besonderen Merkmale und die damit verbundene Integration der impliziten räumlichen Beziehungen (d. h. topologisch, metrisch und gerichtet) und die Eigenschaft der Autokorrelation unter den räumlichen Daten beschrieben.

Spatial Clustering

Das Spatial Clustering entspricht dem Clustering, das in Kapitel 2 beschrieben worden ist, in räumlichen Datenbanken [SC03] [HK00]. Der Grund liegt darin, dass sich ähnliche räumliche Objekte aufgrund der Autokorrelation im Raum eher zu Cluster gruppieren anstatt sich zufällig anzuordnen.

Das Clustering der räumlichen Objekte wird anhand einer Qualitätsfunktion (siehe Kapitel 2), die auf einer Abstandsfunktion basiert, durchgeführt und fasst die Objekte zu Cluster zusammen. Daher ist der räumliche Aspekt essentiell beim Clustering.

Die Methoden zum Clustering, d. h. Partitioning-based, Hierarchical-based, Density-based und Grid-based (siehe Kapitel 2.6.2) können zum Clustering in räumlichen Datenbanken verwendet werden. Jedoch kann es notwendig sein, die geometrischen Datentypen, z. B. den Punkt, in Standardwerte umzuwandeln, d. h. in die Koordinaten x und y. Komplexere räumliche Datentypen, z. B. Linien oder Polygone, lassen sich hingegen nicht ohne weiteres mit den klassischen Clustering-Methoden verarbeiten, da eine geeignete Repräsentation durch Standard-Datentypen nicht gegeben ist.

Räumliche Klassifikation und Regression

Die Klassifikation, die in Kapitel 2 vorgestellt worden ist, beschreibt den Prozess der Klassifizierung eines Objektes (Datensatz einer Tabelle) zu einer Klasse aus einer vorgegebenen Menge von Klassen anhand der Attribute des Objektes. Bei der Räumlichen Klassifikation werden darüber hinaus die Attribute der räumlich benachbarten Objekte (z. B. die nicht-räumlichen Attribute Bevölkerungsdichte und Kriminalitätsrate) und räumliche Beziehungen zwischen den Objekten (z. B. in der Umgebung befindet sich ein See oder eine Schnellstraße) in den Prozess der Klassifizierung einbezogen.

Bei den Objekten in der Räumlichen Klassifikation handelt es sich demnach um Objekte, die in ihre Umgebung eingebettet und folglich als komplexe Objekte zu verstehen sind. Diese erweiterte Auffassung des Objektes ist besonders bei der Räumlichen Klassifikation von Bedeutung.

Beispiel: Klassifizierung einer Region nach *wohlhabend* oder *arm*

In Abb. 5.2 ist ein Ausschnitt einer Karte und die dazugehörige Repräsentation in Form einer Tabelle zu sehen. Die Aufgabe der Räumlichen Klassifikation besteht darin, die unbekannte Region 5 nach *wohlhabend* oder *arm* zu klassifizieren.

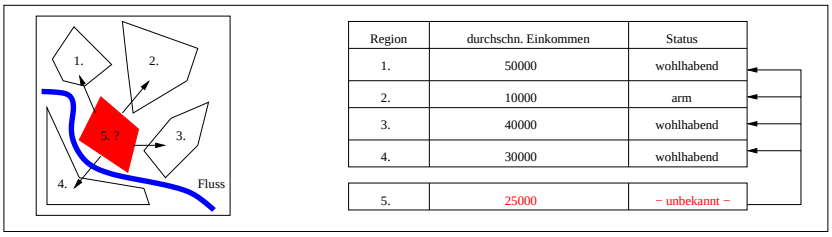


Abbildung 5.2: Integration impliziter Beziehungen

Für die Klassifikation wird ein Klassifikationsmodell benötigt, das sowohl die Attribute des Objektes selbst (z. B. `durchschnittliches Einkommen`) als auch die Attribute der benachbarten Objekte (Stichwort: Autokorrelation), sowie räumliche Beziehungen in den Prozess der Klassifizierung vereinigt.

Bei der Klassifizierung der Region 5 – *wohlhabend* oder *arm* – könnten beispielsweise folgende Betrachtungen in den Entscheidungsprozess einbezogen werden:

- Aggregation des Attributs `durchschnittliches Einkommen` der benachbarten Objekte, z. B. Durchschnittswert (= 32500) oder Maximalwert (= 50000)
- Klassifizierung der benachbarten Objekte (dreimal *wohlhabend* und einmal *arm*)
- Existenz eines Flusses in der unmittelbaren Umgebung

Bei der klassischen Klassifikation wird die Region 5, wobei lediglich das Attribut `durchschnittliches Einkommen` = 25000 in den Klassifikationsprozess einbezogen wird, mit Status=*'arm'* klassifiziert. Durch die erweiterte Betrachtung, d. h. der räumlich benachbarten Objekte und der räumliche Beziehungen, könnte die Region 5 jedoch mit Status= *'wohlhabend'* klassifiziert werden. (Zweite ist die richtige Klassifizierung für Region 5.)

Räumliche Assoziationsanalyse

Das Data Mining von Assoziationsregeln mit der Assoziationsanalyse wird in räumlichen Datenbanken zur Räumlichen Assoziationsanalyse und zu Räumlichen Assoziationsregeln erweitert. Eine Räumliche Assoziationsregel ist eine Regel, die Beziehungen zwischen räumlichen und nichträumlichen Attributen eines geographischen Objektes in räumlichen Datenbanken beschreibt. Zum Beispiel repräsentiert die Regel – „ Große Städte in Deutschland liegen im Westen.“– eine Räumliche Assoziationsregel. [KH95]

Das folgende Beispiel stellt eine konkrete Räumliche Assoziationsregel mit einem Support $s = 5\%$ und einer Confidence $c = 90\%$ dar. Die Regel beschreibt, das 90% der großen Städte, die am Meer gelegen sind, einen Hafen besitzen. Das Prädikat *in_der_Nähe_von* beschreibt dabei eine metrische Beziehung.

$$\texttt{ist\_eine(X, 'große Stadt')} \wedge \texttt{in\_der\_Nähe\_von(X, 'Meer')} \Rightarrow \texttt{besitzt(X, 'Hafen')}$$

Neben dem verwendeten Prädikat *in_der_Nähe_von*, können eine Vielzahl von räumlichen Prädikaten, die aus den räumlichen Beziehungen abgeleitet sind, in einer Räumliche Assoziationsregel enthalten sein.

Anmerkung: Eine Regel wird Räumliche Assoziationsregel genannt, wenn mind. ein Prädikat in der Prämisse oder Konklusion ein räumliches Prädikat ist, ansonsten handelt es sich um eine *normale* Assoziationsregel. [KH95]

Co-Location Analyse

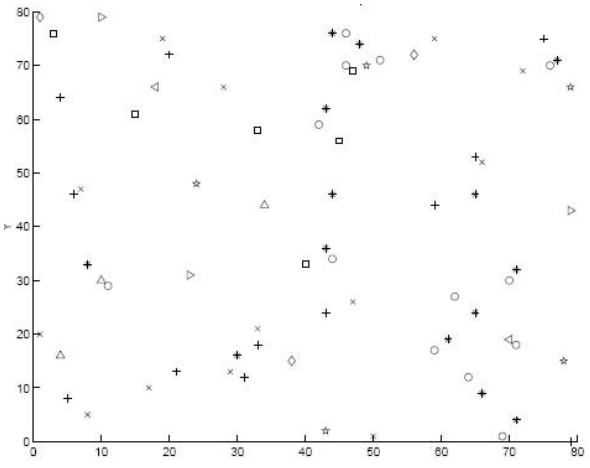


Abbildung 5.3: Beispiel: Co-Location[HPX06]

Bei der Co-Location Analyse wird nach Spatial Co-Location Regeln gesucht. Diese beschreiben ein mehrfaches Vorkommen von räumlichen Eigenschaften, sogenannten *spatial features*, in einer räumlichen Umgebung. In Abb. 5.3 ist beispielsweise ein zwei-dimensionaler Raum dargestellt, in dem sowohl die *spatial features* {'+'} und {'x'}, sowie die *spatial features* und { 'o'} und {'*'} – räumlich betrachtet – gemeinsam auftreten. Im geographischen Kontext repräsentiert ein *spatial feature* bei der Co-Location Analyse ein geographisches Objekt, das entweder vorhanden oder nicht vorhanden ist, im Bezug auf eine Position im Raum, z. B. symbiotische Lebensgemeinschaften – Clownfisch und Seeanemone – in der Ökologie. Diese Co-Location der beiden Organismen ist in der nachfolgenden Co-Location Regel dargestellt, wobei der räumliche Kontext zu beachten ist, in der die Regel eingebettet ist. Der Aufbau einer Co-Location Regel ist dabei an die Form der Assoziationsregeln orientiert. Die Regel besagt: Wenn die Seeanemone ({'+'}) im Ozean an der Position P vorhanden ist, dann ist ebenfalls der Clownfisch ({'x'}) an der Position P vorhanden.

$$\texttt{Seeanemone} \Rightarrow \texttt{Clownfisch}.$$

Dabei ist die Co-Location Analyse mit der Assoziationsanalyse vergleichbar. Tatsächlich gibt eine Vielzahl von Übereinstimmungen, z. B. ein *item* in der Assoziationsanalyse entspricht einem *spatial feature* in der Co-Location Analyse und eine Menge von *items* entspricht einer Menge von *spatial features*. Jedoch unterscheiden sich die Co-Location Analyse und die Assoziationsanalyse in den folgenden entscheidenden Punkten:

- Der Begriff der Transaktion in der Assoziationsanalyse wird bei der Co-Location Analyse durch den Begriff der Umgebung ersetzt, da die Daten in einem kontinuierlichen Raum (d. h. nicht diskretisiert) eingebettet sind.
- Der Begriff des Support bei der Assoziationsanalyse ist für die Bewertung einer Co-Location Regel nicht ausreichend, da die Anzahl der *spatial features* im Vergleich zur Gesamtmenge der räumlichen Objekte sehr klein sein kann. [SC03]

Obwohl die Co-Location Analyse und die Assoziationsanalyse sich in einigen Aspekten unterscheiden, kann die Co-Location Analyse in eine geeignete Assoziationsanalyse umgewandelt werden. Dazu existieren zwei verschiedene Herangehensweisen: [SC03] [HPX06] [Mor01]

1. *reference-feature centric*: Zunächst wird ein *spatial feature* als Referenz-Objekt ausgewählt und anschließend werden weitere *spatial features*, die sich in einer definierten Umgebung um das Referenz-Objekt befinden, zu einer Transaktion zusammenfasst. Die Anzahl der Transaktionen wird dabei durch die Anzahl der ausgewählten Referenz-Objekte definiert.
2. *data partitioning*: Dabei wird der Datenraum in Partitionen (Grid-Muster) unterteilt und anschließend die *spatial features* der jeweiligen Partition zugeordnet. Eine Transaktion entspricht hierbei einer Partition. Die Anzahl der Transaktionen wird durch die Partitionierung vorgegeben.

Räumliche Trend Analyse

Die Räumliche Trend Analyse ist eine Data Mining-Methode, die ausschließlich in räumlichen Datenbanken Anwendung findet, und untersucht Änderungen und Tendenzen in den Daten entlang einer räumlichen Dimension (siehe Abb. 5.4 rechts). Bei der Analyse wird nach Tendenzen sowohl in nicht-räumlichen als auch räumlichen Attribute gesucht. Ein Beispiel für einen Trend wird durch folgende Aussage (Regel) beschrieben: Wenn man sich in nordwestlicher Richtung vom Zentrum einer Metropole wegbewegt, dann steigt das durchschnittliche Einkommen der Bevölkerung (Confidence = 80%).

In Abb. 5.4 (links) ist das Ergebnis der Räumlichen Trend Analyse anschaulich anhand von zwei Beispielen graphisch dargestellt. Im ersten Beispiel handelt es sich um einen sogenannten *globalen Trend*, d. h. es ist eine ähnliche Änderung in nicht-räumlichen oder räumlichen Attributen ausgehend von einem Startpunkt in jede beliebige Richtung zu beobachten (z. B. Analogie zum Berggipfel). Im zweiten Beispiel handelt es sich um einen sogenannten *lokalen Trend*, d. h. die Art der Änderung in nicht-räumlichen oder räumlichen Attributen verhält sich unterschiedlich im Bezug auf die betrachtete Richtung.

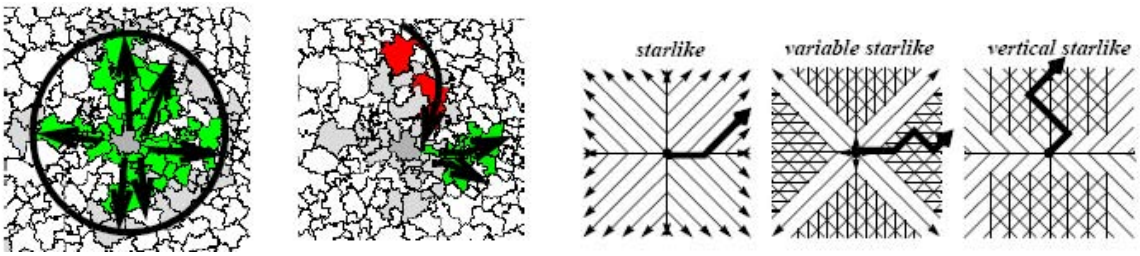


Abbildung 5.4: Beispiel: Räumliche Trend Analyse und Definition der Suchpfade [EFKS98]

5.2.2 Lösungsansätze für das Spatial Data Mining

Im Folgenden werden Ideen und Lösungsansätze präsentiert, die in der Literatur vorgeschlagen werden, um Spatial Data Mining in räumlichen Datenbanken zu verwirklichen. Zunächst werden

Ideen vorgestellt, die jeweils für eine einzelne Methode gedacht sind. Anschließend werden komplette Lösungsansätze, sogenannte *Frameworks*, vorgestellt, mit denen möglichst viele klassische Data Mining-Methoden für die Verwendung zum Spatial Data Mining angepasst und erweitert werden sollen. Außerdem sollen Ideen zur Lösung der Methoden, die ausschließlich in räumlichen Datenbanken Anwendung finden, d. h. Räumliche Trend Analyse und Co-Location Analyse, vorgestellt werden.

Lösungsmethoden zum Spatial Clustering

Da die bisher vorgestellten Clustering-Methoden (siehe Kapitel 2.6.2) lediglich Standard-Datentypen umgehen können, sind Methoden entwickelt worden, die auch komplexere Datentypen, z. B. Polygone umgehen können. Eine Lösung für das Clustering von beliebigen geometrischen Objekten, z. B. Polygonen, wird von [SEKX98] beschrieben.

In [SEKX98] wird der GDBSCAN-Algorithmus (Generalized Density Based Spatial Clustering of Application with Noise) vorgestellt, der auf der Density-based Methode basiert und eine Verallgemeinerung des DBSCAN-Algorithmus für Punktmengen ist (siehe Kapitel 2.6.2).

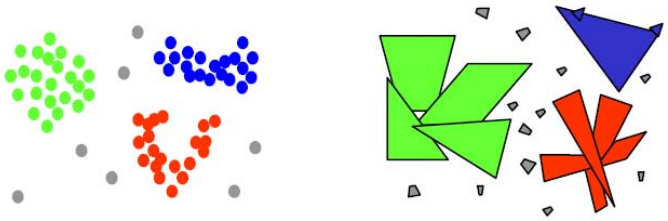


Abbildung 5.5: Spatial Clustering [SEKX98]

Die Art der Verallgemeinerung umfasst dabei die Definition der Dichte und der Nachbarschaft. Im DBSCAN-Algorithmus ist die Dichte durch eine Mindest-Anzahl an Punkten und die Nachbarschaft durch eine ϵ-Umgebung um einen Punkt gegeben. Die Angaben sind für das Clustering von Punkten nach der Density-based Methode essentiell.

Um das Clustering von beliebigen geometrischen Objekten, speziell von Polygonen, zu ermöglichen, ist eine Neudefinition der Begriffe Dichte und Nachbarschaft notwendig. In Abb. 5.5 sind zum einen drei Cluster zu identifizieren und zum anderen Ausreißer (graue Polygone), die keinem Cluster zugeordnet sind. Die Nachbarschaft wird dabei über die topologische Beziehung '*schneidet*' und die Dichte über die Größe der Schnittfläche definiert. D. h. erfüllen zwei Polygone die topologische Beziehung '*schneidet*' und ist außerdem die Schnittmenge größer als ein vorgegebener Wert, so sind diese zueinander ähnlich und formen somit einen Cluster.

Der Algorithmus und die Implementierung des GDBSCAN-Algorithmus sind in [SEKX98] beschrieben. Außerdem wird die Anwendung des Algorithmus anhand realer Beispiele demonstriert, z. B. Geographie (2D Polygone), Astronomie (2D Punkte), Biologie (3D Punkte) und Geo-Wissenschaften (5D Punkte).

Lösungsmethoden zur Räumlichen Klassifikation und Regression

In [KHS] wird eine Methode zur Konstruktion von Klassifikationsbäumen vorgestellt, die die Klassifikation von Daten mit räumlichem Bezug ermöglicht. Der vorgestellte Algorithmus basiert auf dem ID3-Algorithmus und verwendet zur Konstruktion des Klassifikationsbaums sowohl nichträumliche Attribute (z. B. Bevölkerung einer Stadt), als auch räumliche Attribute in Form von Prädikaten (z. B. Bevölkerung im Umkreis von 2 km einer Stadt).

Des Weiteren wird eine Selektion der relevanten Attribute für den Klassifikationsprozess vorgeschlagen, die auf dem Konzept des *nearest hit* (dichtester Nachbar gehört zur gleichen Klasse) und

nearest miss (dichtester Nachbar gehört zu einer anderen Klasse) basiert. Die Methode verarbeitet dabei zu jedem Objekt, umgeben von einem definierten Bereich (Puffer), die räumlichen Beziehungen und legt die Ergebnisse bezüglich jedes Objektes in Form von Prädikaten, z. B. `durchschnittliches_Einkommen(X, 'hoch')`, in einer Tabelle ab. Dabei besteht ein wichtiger Zwischenschritt dieser Methode aus der effizienten Vorverarbeitung der räumlichen Beziehungen, die sich entsprechend in zwei Phasen unterteilt: In der ersten Phase wird eine grobe und ungenaue Berechnung und in der zweiten Phase eine verfeinerte Berechnung für die Ergebnisse der ersten Phase durchgeführt.

Anschließend wird der Klassifikationsbaum auf der relationalen Tabelle mit den nicht-räumlichen und räumlichen Attributen definiert, wobei der Algorithmus derart modifiziert ist, dass die räumlichen Attribute, d. h. die Prädikate, verarbeitet werden können.

In [EFKS98] wird ein anderer Ansatz zur Räumlichen Klassifikation verfolgt. Dieser basiert zwar ebenfalls auf der Erweiterung des ID3-Klassifikations-Algorithmus für relationale Datenbanken, jedoch werden keine Puffer und keine räumlichen Prädikate wie in [KHS] benutzt, sondern es wird ein sogenannter Nachbarschafts-Index (Abb. 5.6) eingeführt, der zu jedem Objekt die topologischen, metrischen, gerichteten Nachbarschaftsbeziehungen speichert. (Weitere Informationen zum Nachbarschafts-Index sind in Abschnitt 5.2.3 *Primitive für Spatial Data Mining* zu finden.)

Object-ID	Neighbour	Distance	Direction	Topology
o_1	o_2	2.7	southwest	disjoint
o_1	o_3	0	northwest	overlap
...	...	...	...	...

Abbildung 5.6: Nachbarschafts-Index in [EFKS98] zur Abbildung der Nachbarschaft

Der Algorithmus zur Konstruktion des Klassifikationsbaums wird demzufolge um die Möglichkeit des Zugriffs und der Auswertung der Nachbarschaft erweitert. Dadurch können bei der Konstruktion des Klassifikationsbaums sowohl die nicht-räumlichen Attribute, die das räumliche Objekt beschreiben, sondern auch die nicht-räumlichen Attribute der Nachbar-Objekte, die über eine Nachbarschaftsbeziehung miteinander verbunden sind, in den Prozess integriert werden.

Der vorgeschlagene Algorithmus in [EFKS98] besitzt jedoch einige Beschränkungen. Zum einen werden keine Konzept-Hierarchien unterstützt, die für nicht-räumliche Attribute definiert sein könnten. Und zum anderen werden aggregierte Werte für nicht-räumliche Attribute ignoriert, z. B. besitzt eine Stadt, in deren Nachbarschaft sich drei Regionen mit einer mittleren Bevölkerungszahl befinden, die gleichen Eigenschaften wie eine Stadt, in deren Nachbarschaft sich nur eine einzelne Region mit einer hohen Bevölkerungszahl befindet[KHS].

In [RT04] wird eine weitere Methode zur Klassifikation von räumlichen Daten mit einem Klassifikationsbaum-Algorithmus vorgestellt, der ebenfalls auf dem klassischen ID3-Algorithmus basiert. Im Gegensatz zu [KHS] und den vorgeschlagenen Puffern und räumlichen Prädikaten, und zu [EFKS98] mit der Abbildung der Nachbarschaftsbeziehungen über den Nachbarschafts-Index zur Realisierung der Räumlichen Klassifikation, wird in diesem Ansatz das Prinzip des Schichten-Modells (siehe Abb. 5.8 bei der Räumlichen Assoziationsanalyse) verwendet. Dabei wird in [RT04] versucht die Begrifflichkeiten und das Vorgehen des klassischen ID3-Algorithmus in relationalen

Datenbanken, d. h. der Datensatz, das Klassifikationsattribut und die Attribute (Prädikatoren für die Klassifizierung), für die Räumlichen Klassifikation in räumlichen Datenbanken abzubilden.

Der Datensatz entspricht einer Karte, die sich aus mehreren Schichten zusammensetzt. Und die Attribute entsprechen den Schichten, wobei eine Schicht jeweils eine Menge von geographischen Eigenschaften darstellt. Z. B. kann eine Karte aus drei Schichten bestehen: 1. Schicht mit Städten, 2. Schicht mit Flüssen und Seen und 3. Straßen. Da für die Konstruktion des Klassifikationsbaums mit derartigen Definitionen für Datensatz und Attribut keines der bekannten Aufteilungskriterien eingesetzt werden kann, wird ein Aufteilungskriterium, der *spatial information gain*, eingeführt, das auf der Entropy (siehe Kapitel 2.6.4) basiert und den räumlichen Aspekt berücksichtigt.

Die eigentliche Konstruktion des Klassifikationsbaums erfolgt analog zur Konstruktion nach der klassischen Methode. Zunächst wird ein Klassifikationsattribut, d. h. eine Schicht (z. B. Städte), ausgewählt. Anschließend wird die Datenmenge – anhand des *spatial information gain*-Aufteilungskriteriums – sukzessive unterteilt, bis eine Abbruchbedingung erreicht ist.

Ein Nachteil der Methode in [RT04] ist es, dass lediglich Schichten verarbeitet werden können, deren geometrische Objekte ausschließlich Polygone sind. Der Grund liegt in der Berechnung des *spatial information gain*-Aufteilungskriteriums, da dieser keine Linien oder Punkte bewerten kann.

In [SC03] wird ein komplett gegensätzlicher Ansatz zu den bisher beschriebenen Methoden für die Räumliche Klassifikation verfolgt. Es wird die klassische Klassifikation durch eine Anpassung des statistischen Modells, insbesondere durch die Erweiterung um die Betrachtung der räumlichen Autokorrelation, zur Räumlichen Klassifikation erweitert. Diese Erweiterung wird anhand der linearen Regression demonstriert, die in [SC03] Spatial Autoregressive Regression (SAR) genannt wird. Im Folgenden wird zum einen eine lineare Regressionsfunktion und zum anderen dieselbe lineare Regressionsfunktion mit einer Erweiterung um die räumliche Autokorrelation gezeigt. Der Einfluss der Autokorrelation wird dabei durch die Nachbarschaftsmatrix W (siehe Abb. 5.7) und den Korrelationskoeffizienten ρ beschrieben. Ist dieser Koeffizient 0, so ergibt sich die klassische Regressionsfunktion.

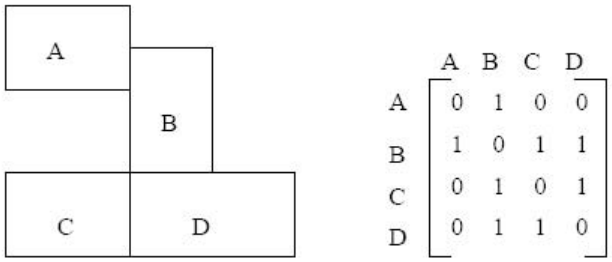


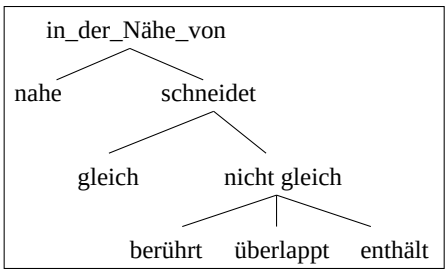
Abbildung 5.7: Nachbarschaftsmatrix W [SC03]

$$f(X) = Y = \alpha \cdot X + \beta \qquad \rightarrow \qquad Y = \rho \cdot W \cdot Y + \alpha \cdot X + \beta$$

Die Berechnung dieser SAR-Funktion ist im Vergleich zur klassischen linearen Regressionsfunktion aufgrund des Terms $\rho \cdot W \cdot Y$ wesentlich komplexer. Jedoch beschreibt das modifizierte Modell die räumlichen Daten durch Einbeziehung der statistischen Abhängigkeit und der Autokorrelation bedeutend besser.

Lösungsmethoden zur Räumlichen Assoziationsanalyse

Bei [KH95] handelt es sich um einen der ersten Ansätze, das allgemeine Konzept der Assoziationsanalyse auf räumliche Datenbanken zu erweitern. Darin wird eine Methode vorgestellt, die Räumliche Assoziationsregeln ermittelt, wobei Assoziationen zwischen Objekten, basierend auf einer räumlichen Beziehung, beschrieben werden. Bei der vorgeschlagenen Methode wird das Prinzip der Referenz-Schicht benutzt, wobei deren Objekte, sogenannte Referenz-Objekte, in Bezug zu Objekten in anderen Schichten stehen. Durch die *Verschneidung* der Referenz-Schicht (z. B. Städte) mit jeweils einer weiteren Schicht (z. B. Straßen, Flüssen und Parkanlagen) werden unter Verwendung einer räumlichen Beziehung (z. B. *in_der_Nähe_von*) Assoziationsregeln ermittelt.



Die Methode selbst beschreibt ein zweistufiges Top-Down--Verfahren, das in der ersten Stufe eine grobe Analyse durchführt und mit dem resultierenden Ergebnis in der zweiten Stufe eine feinere Analyse folgen lässt. Damit die Analyse nach der Methode in [KH95] in den beschriebenen zwei Stufen durchgeführt werden kann, wird zumindest eine Konzept-Hierarchie für die räumlichen Prädikate benötigt, z. B. für topologische Beziehungen (Abb. rechts). Eine Konzept-Hierarchie für nicht-räumliche Prädikate ist optional, jedoch könnten damit Multi-level-Assoziationsregeln (siehe Kapitel 2) gefunden werden.

Die Arbeitsweise der Methode in [KH95] lässt sich wie folgt beschreiben, wobei die folgenden Parameter spezifiziert werden müssen: a) räumliche Datenbank mit räumlichen und nicht-räumlichen Attributen, b) (räumliche und nicht-räumliche) Konzept-Hierarchien, c) eine Referenz-Schicht, eine Menge von relevanten thematischen Karten und eine räumliche Beziehung, sowie d) Minimum-Support und Minimum-Confidence :

1. Im ersten Schritt werden *frequent itemsets* bei einem hohen Abstraktionsgrad (hohes Level in der Konzept-Hierarchie) gesucht. Dabei beschreibt bei dieser Methode ein *frequent itemset* eine Menge von *items*, die wiederum räumlichen Prädikaten (siehe Abschnitt 5.2.1) entsprechen. In diesem Schritt werden potentielle Kandidaten, die die spezifizierte räumliche Beziehung erfüllen könnten, identifiziert, wobei effiziente räumliche Algorithmen und Datenstrukturen, wie z. B. R-Bäume und *Plane-Sweep*-Techniken, verwendet werden.
2. Anschließend werden die *frequent itemsets* im zweiten Schritt einer feineren Analyse durch Anwendung rechenintensiver räumlicher Algorithmen, wie z. B. *Spatial Joins* oder Operationen für topologische Beziehungen, unterzogen. Außerdem wird in diesem Schritt der Abstraktionsgrad sukzessive reduziert (niedriges Level in der Konzept-Hierarchie), z. B. `in_der_Nähe_von` geht über zu `schneidet` und `nahe` oder `Straße` geht über zu `Autobahn` und `Bundesstraße`. Der Prozess wird solange durchgeführt bis entweder das unterste Level der Konzept-Hierarchie erreicht ist oder nur noch einelementige *frequent itemsets* gefunden werden.

Beispiel einer Räumlichen Assoziationsanalyse nach [KH95]:

> Gesucht sind Assoziationen in Form von Räumlichen Assoziationsregeln zwischen Städten in *British Columbia* mit Straßen, Gewässer, Minen oder Grenzen mit einem vorgegebenen Support von 5% und einer Confidence von 80%.
> Dadurch könnte beispielsweise die nachfolgende Assoziationsregel, unter Einbeziehung der obigen Konzept-Hierarchie entdeckt werden, die beschreibt, dass 80% der Städte in *British Columbia* in der Nähe von Wasser gelegen sind.
>
> $$\texttt{ist(X, 'Stadt')} \Rightarrow \texttt{nahe(X, Y)} \wedge \texttt{ist(Y, 'Gewässer')}\ [\text{s=40\%, c=80\%}]$$

Diese Methode ermittelt Räumliche Assoziationsregeln, die das Referenz-Objekt und jeweils ein räumliches Objekt aus der Menge der vorgegebenen Objekte beinhaltet und durch die vorgegebenen Beziehungen in der Konzept-Hierarchie verbindet. Ein erheblicher Nachteil dieser Methode besteht darin, dass zum einen Hintergrundwissen in Form von Konzept-Hierarchien verlangt wird und zum anderen vom Benutzer erwartet wird, dass er genau die Art der Assoziationsregel vorgibt, die gefunden werden sollen, d. h. Referenz-Objekt, eine Menge von räumlichen Objekten und eine räumliche Beziehung, die zwischen den Objekten gelten soll.

Eine weitere Methode zur Räumlichen Assoziationsanalyse wird in [RT05] beschrieben. Diese erweitert den Ansatz von [KH95] dahingehend, dass nicht nur jeweils zwei Schichten miteinander

Verschnitten werden, sondern beliebig viele Schichten. Für diesen Ansatz werden die Begriffe der *räumlichen Transaktionen* und der *räumlichen Items* in Analogie zur Warenkorb-Analyse [AMS+96] eingeführt.

Mit der vorgeschlagenen Methode werden Räumliche Assoziationsregeln ermittelt, die Assoziationen zwischen Objekten, basierend auf einer räumlichen Beziehung, beschreiben. Dabei wird ebenfalls das Prinzip der Referenz-Schicht benutzt, wobei deren Objekte, die Referenz-Objekte, die *räumlichen Transaktionen* für die Assoziationsanalyse vorgeben. Die *räumlichen Items* einer *räumlichen Transaktion* werden dabei durch die *Verschneidung* der Referenz-Schicht (z. B. Städte) mit weiteren Schichten (z. B. Straßen, Flüssen und Parkanlagen) unter Verwendung einer räumlichen Beziehung (z. B. *'schneidet'*) bestimmt. Die folgende Anweisung *verschneidet* drei Schichten unter der topologischen Beziehung *'schneidet'*, wobei die erste Schicht die Referenz-Schicht ist:

```
SELECT l1.*, l2.*, l3.* FROM layer 1 as l2, layer 2 as l2, layer 3 as l3
WHERE schneidet(l1.geometry, l2.geometry) = 'TRUE'
AND schneidet(l1.geometry, l3.geometry) = 'TRUE'
```

Anschaulich beschreibt Abb. 5.8 eine *räumliche Transaktion*, die eine Menge von nicht-räumlichen Attributen (z. B. Bevölkerung, Niederschlag, etc.) aus den überlagerten Schichten repräsentiert, die bezogen auf das Referenz-Objekt in der Referenz-Schicht in allen Schichten gültig sind.

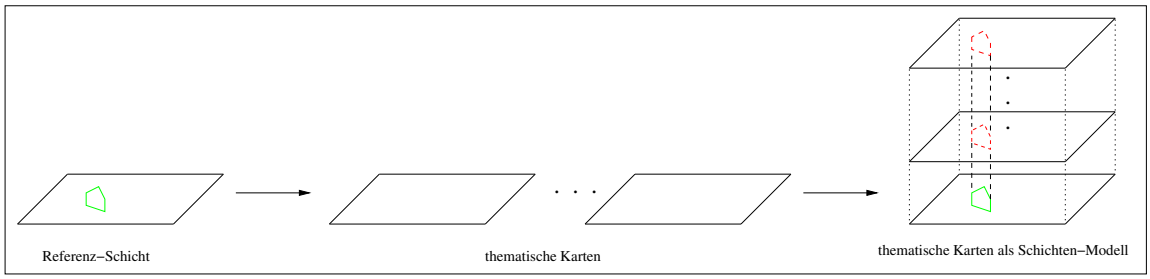


Abbildung 5.8: Definition einer *räumlichen Transaktion* nach [RT05]

Beispiel:

Es seien folgende drei thematische Karten gegeben: 1. *Städte*, 2. *Straßen* und 3. *Beförderungsmittel*. Außerdem seien geeignete Konzept-Hierarchien für die nicht-räumlichen Attribute der Schichten gegeben. Durch die Überlagerung der thematischen Karten nach dem Prinzip in Abb. 5.8 und unter Verwendung der topologischen Beziehung *'schneidet'*, werden die räumlichen Transaktionen konstruiert, wobei die Schicht *Städte* die Referenz-Schicht darstellt. Dann werden beispielsweise folgende Assoziationsregeln mit der Räumlichen Assoziationsanalyse bestimmt, wobei X eine Region oder Stadt in der Referenz-Schicht *Städte* repräsentiert:

$$\texttt{schneidet(X, 'Autobahn')} \wedge \texttt{Einwohner(X, '>500000')} \Rightarrow \texttt{enthält(X,'Flughafen')}$$

oder

$$\texttt{schneidet(X, 'Fluss')} \wedge \texttt{schneidet(X, 'Landstraße')} \Rightarrow \texttt{Einwohner(X, '<5000')}$$

Die Assoziationsregeln beschreiben Assoziationen, die nicht nur auf zwei Schichten – Referenz-Schicht und eine weitere thematische Karte – beschränkt ist. Des Weiteren sind Assoziationsregeln denkbar, die die Referenz-Schicht ausschließlich zur Definition der *räumlichen Transaktion* benötigen, aber kein eigentlicher Bestandteil der Regel ist.

Lösungsmethoden zur Spatial Co-Location Analyse

In [HPX06] werden gleich zwei Methoden vorgeschlagen, mit denen Co-location Regeln ermittelt werden. Eine Transaktion wird dabei anschaulich durch einen Nachbarschaftsgraphen beschrieben, wobei zwei Objekte jeweils verbunden sind, wenn diese einen max. Abstand nicht überschreiten. Die Besonderheit dieser Methoden besteht darin, dass selbst Co-location Regeln gefunden werden, deren Vorkommen nicht so häufig ist.

Die erste Methode beschreibt eine Erweiterung der Assoziationsanalyse-Methode aus [AMS+96], wobei zunächst ein support-ähnliches Maß (*maximal participation index*) zur Einschränkung des Suchraums benutzt wird und anschließend in einem Postprocessing ein confidence-ähnliches Maß eingesetzt wird, um die Co-location Regeln zu ermitteln. Diese Methode hat jedoch den Nachteil, dass bei zu groß gewähltem Support nicht alle Co-location Muster gefunden werden und bei einem zu kleinen Support zwar alle gefunden werden, jedoch überdurchschnittlich viele Muster zur Analyse verarbeitet werden. Die zweite Methode hingegen macht sich eine spezielle Monotonie-Eigenschaft der Co-Location Muster zu nutze, um den Suchraum – ähnlich wie die Monotonie-Eigenschaft beim Apriori-Algorithmus [AMS+96] – einzuschränken und dennoch alle (auch *seltenen*) Muster findet.

In [Mor01] wird eine weitere Methode für die Co-Location Analyse beschrieben, wobei das Ziel dieser Methode das Finden von sogenannten *Neighboring Class Sets* ist. Ein *Neighboring Class Set* beschreibt dabei eine Menge von unterschiedlichen Objekten, die sich räumlich betrachtet gruppieren. Semantisch betrachtet, beschreibt diese Methode demnach eine Co-Location Analyse, da die Co-Location Muster und die *Neighboring Class Sets* synonym sind.

Dabei verwendet [Mor01] einen Apriori-Algorithmus, der sukzessive *frequent-k-Neighboring Class Sets* ($k \geq 2$), ausgehend von *frequent-1-Neighboring Class Sets*, unter Berücksichtigung eines Supports generiert. Der Algorithmus benutzt zur Bestimmung und zur Bewertung von Nachbarschaften ein Voronoi-Diagramm, das eine Datenstruktur zur Speicherung von Nachbarschaften darstellt. Das Vorgehen zur Findung von *frequent-(k+1)-Neighboring Class Sets* aus *frequent-k-Neighboring Class Sets* erfolgt dabei durch Generierung von $(k+1)$-Kandidat-Mengen aus jeweils zwei k-Mengen, wobei sich die k-Mengen lediglich in einem Objekt unterscheiden dürfen. Anschließend wird unter Verwendung des Voronoi-Diagramms die $(k+1)$-Kandidat-Menge berechnet und geprüft, ob ein vorgegebener Maximal-Abstand zwischen den Objekten nicht überschritten wird. Ist diese Bedingung und der Minimum-Support für die $(k+1)$-Kandidat-Menge erfüllt, so handelt es sich um ein *frequent-(k+1)-Neighboring Class Sets*.

Nachteilig bei [Mor01] ist, dass zum einen lediglich Punkt-Darstellungen verarbeitet werden können und zum anderen zu viele uninteressante *frequent Neighboring Class Sets* generiert und analysiert werden müssen.

5.2.3 Frameworks für Spatial Data Mining

Die nun folgenden Lösungsansätze beschreiben, im Gegensatz zu den Lösungsansätzen im vorvergangenen Abschnitt, komplette *Frameworks*, deren Motivation es ist, die gesamten existierenden Problemstellungen des Spatial Data Mining durch universelle Konzepte zu begegnen.

Primitive für Spatial Data Mining

In [EKS97] wird eine allgemeine Erweiterung der Data Mining-Methoden vorgeschlagen, die eine Integration von Nachbarschaftsbeziehungen in Form von Nachbarschaftsgraphen und entsprechenden Operationen für den Graphen anstreben, um damit Knowledge Discovery in räumlichen

Datenbanken durchführen zu können. Das Ziel ist die Verarbeitung von impliziten räumlichen Beziehungen und räumlichen Nachbarschaften durch das DBMS.

Die Basis-Operationen (*Primitives*) basieren auf den impliziten räumlichen Beziehungen (d. h. topologisch, metrisch, gerichtet) und der Annahme, dass die Data Mining-Algorithmen für räumliche Datenbanken diese nutzen. Des Weiteren wird die Abbildung der impliziten Nachbarschaftsbeziehungen durch das Konzept des Nachbarschaftsgraphen und der Nachbarschaftspfade realisiert. Die Konstruktion des Nachbarschaftsgraphen erfolgt durch die Definition der Nachbarschaft anhand einer Nachbarschaftsbeziehung (z. B. topologische Beziehung - *schneidet*).

Ein Nachbarschaftsgraph bezüglich einer räumlichen Beziehung stellt einen Graphen dar, dessen Knoten Objekte in der Datenbank und dessen Kanten durch die räumliche Beziehung verbundene Objekte repräsentieren. Und ein Nachbarschaftspfad beschreibt einen Pfad, der zyklenfrei und räumlich betrachtet in eine einmal festgelegte Richtung geht, d. h. die Pfade breiten sich sternförmig von einem Zentrum aus.

Die Basis-Operationen, die auf dem Nachbarschaftsgraphen operieren, sind im Folgenden auflistet:

- *get-Graph* – Erstellt einen Nachbarschaftsgraphen bezüglich einer räumlichen Beziehung
- *get-Neighborhood* – Ermittelt alle direkt-verbundenen Nachbar-Objekte, die einer vorgegebenen Bedingung genügen
- *create-Path* – Erstellt eine Menge von Nachbarschaftspfaden ausgehend von einem spezifizierten Objekt
- *extend-Path* – Erweitert einen Nachbarschaftspfad um eine vorgegebene Anzahl an Kanten

Für eine effiziente Unterstützung der Primitive und des Nachbarschaftsgraphen wird der Nachbarschafts-Index eingeführt. In Abb. 5.9 ist beispielhaft ein Nachbarschafts-Index dargestellt. Dieser kann durch die räumlichen Index-Strukturen, z. B. B+ -Baum oder R-Baum, realisiert werden.

Object-ID	Neighbour	Distance	Direction	Topology
o_1	o_2	2.7	southwest	disjoint
o_1	o_3	0	northwest	overlap
...	...	...	...	...

Abbildung 5.9: Nachbarschafts-Index in [EFKS98]

In [EKS97] wird weiterhin gezeigt, dass Spatial Data Mining-Algorithmen mit den vorgeschlagenen Primitiven realisiert werden können. Als Beispiel für die Räumliche Assoziationsanalyse wurde die Methode von [KH95] untersucht und der Ablauf der Methode derart modifiziert, dass die Berechnung der *Spatial Joins* der räumlichen Daten durch die Primitive durchgeführt wird. Ein weiteres Beispiel wird mit der Spatial Klassifizierung und der Methode von [KHS] angeführt, bei dem die in der Methode verwendeten räumliche Bereiche um ein Objekt herum, sogenannte Buffer, durch die Primitive und den Nachbarschaftsbeziehungen erzeugt werden.

Anmerkung: Die Räumliche Trend Analyse, die ausschließlich in räumlichen Datenbanken angewendet werden kann, wird in natürlicher Weise und **ausschließlich** durch den vorgeschlagenen Nachbarschaftsgraphen und Nachbarschaftspfade unterstützt.

Materialisierung – Ein Framework

In [VB05b] wird der Knowledge Discovery-Prozess, der in Kapitel 2 vorgestellt worden ist, für die Anwendung in räumlichen Datenbanken modifiziert. Der Ablauf des KDD-Prozesses in räumlichen Datenbanken ist in Abb. 5.10 abgebildet. Dabei sollen die eingefügten Schritte nach [VB05b] in einem Framework gekapselt werden, mit dem dann beliebige räumliche Datenbanksysteme (SDBMS) und Data Mining-Implementierungen arbeiten können. Eine erste Implementierung ist in [VB06a] beschrieben, in der das SDBMS PostGIS und die freie Data Mining-Bibliothek WEKA [WEK] verwendet wird.

Die Erweiterung (Abb. 5.10) des Knowledge Discovery-Prozesses umfasst die folgenden Schritte:

1. Selektion (*Selection*) – In diesem Schritt werden die relevanten Daten für die Knowledge Discovery ausgewählt. Dieser Schritt unterteilt sich dabei in zwei Unterschritte – 1. Daten-Definition und 2. Anwendung von nicht-räumlichen Filtern. Die Daten-Definition beschreibt die Auswahl der Relationen und der Attribute der Datenbank, die für die weitere Verarbeitung relevant sind. Und die nicht-räumlichen Filter geben Bedingungen vor, die die Daten, die durch die Daten-Definition spezifiziert sind, erfüllen müssen.
2. Materialisierung (*Materialization*) – Dieser Schritt, unterteilt in die Unterschritte *Spatial Join* und *Spatial Optimizer*, materialisiert (transformiert) die räumlichen Daten, d. h. alle räumlichen Datentypen und räumlichen Beziehungen (topologisch, metrisch, gerichtet) werden zu nicht-räumlichen Attributen umgewandelt. Für die Materialisierung werden dabei Optimierungsfunktionen und Indexe zur Beschleunigung der räumlichen Berechnungen definiert.
3. Transformation (*Transformation*) – In diesem Schritt werden die nicht-räumlichen Attribute und alle materialisierten Informationen aus dem vorangegangenen Schritt zu einer einzelnen Tabelle zusammengefasst. Diese Tabelle wird anschließend mit dem klassischen Data Mining verarbeitet.

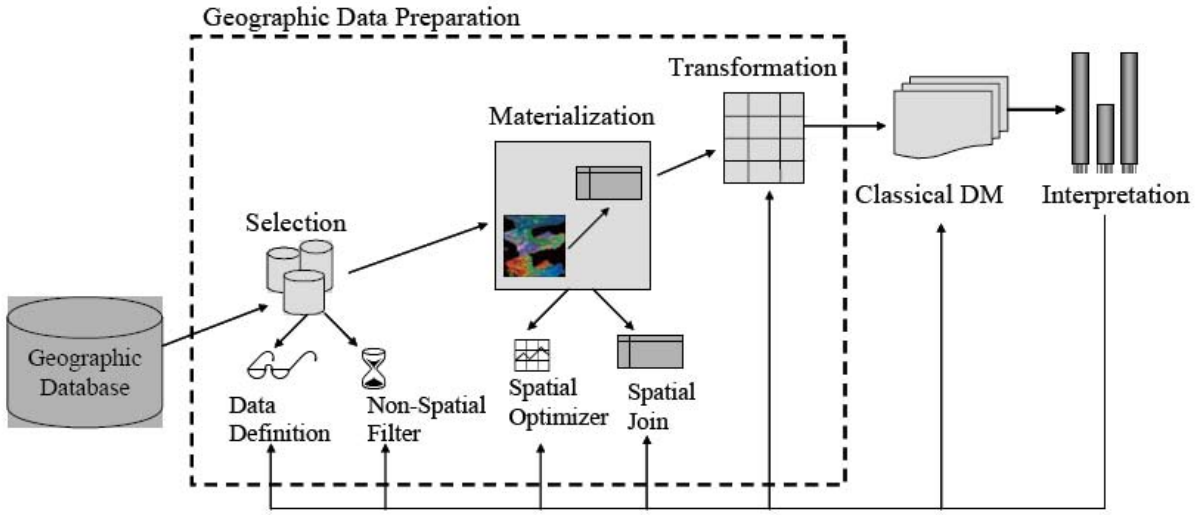


Abbildung 5.10: Knowledge Discovery in räumlichen Datenbanken [VB05b]

Die Materialisierung der räumlichen Beziehungen, beschränkt auf topologische und metrische Beziehungen, wird in [VB06b] beschrieben. Dabei werden zwei Vorgehensweisen bei der Materialisierung vorgeschlagen:

- *feature instance* – beschreibt die Materialisierung einer räumlichen Beziehung nach dem räumlichen *feature* und der Identifizierung durch die *instance*. Z. B. Die Stadt `Hannover` wird vom Fluss `Leine` geschnitten, d. h. topologische Beziehung *schneidet(Hannover, Fluss=Leine) = TRUE*.

Die Materialisierung erfolgt dann durch die Erweiterung der Relation `Städte` um das Attribut `Fluss_Leine` (`Fluss` entspricht *feature* und `Leine` entspricht *instance*), das den Wert *schneidet* erhält. Diese Art der Materialisierung hat zur Folge, dass die Relation für jedes materialisierte Objekt um ein neues Attribut erweitert wird. Des Weiteren wird durch die Beschränkung der topologischen Beziehungen auf das 9-IM [Ege91] gewährleistet, dass keine Mehrdeutigkeiten zwischen dem *feature* und der *instance* auftreten können.

- *feature type* – beschreibt die Materialisierung einer räumlichen Beziehung nach dem räumlichen *feature* und dem *type* der räumlichen Beziehung. Z. B. Die topologische Beziehung *schneidet(Hannover, Fluss=Leine) = TRUE* wird zunächst auf *schneidet(Hannover, Fluss) = TRUE* reduziert und anschließend durch die Erweiterung der Relation `Städte` um ein weiteres Attribut `Fluss_schneidet` (`Fluss` entspricht *feature* und `schneidet` entspricht *type*), das den Wert *TRUE* erhält, materialisiert.

Anmerkung: Unter der Annahme, dass die Anzahl der Objekte größer als die Anzahl der räumliche Beziehungen ist, erzeugt die Materialisierung nach dem Vorgehen mit *feature instance* eine große Anzahl an Attributen, die einen niedrigen Abstraktionsgrad besitzen. Im Gegensatz dazu erzeugt die Materialisierung mit *feature type* weniger Attribute, jedoch mit einem hohen Abstraktionsgrad.

Oracle Spatial Analysis and Mining

Eine weitere Lösung für das Spatial Data Mining bietet die Oracle Spatial-Cartridge mit dem SAM-Package (Spatial Analysis and Mining)[Ora06g] an, wobei die Materialisierung von räumlichen Informationen im Vordergrund steht. Durch die Materialisierung werden die räumlichen Beziehungen und Eigenschaften auf nichträumliche Attribute (Standard-Datentypen) abgebildet, woraufhin anschließend die Data Mining-Cartridge und die darin enthaltenen Methoden für das Data Mining benutzt werden kann.

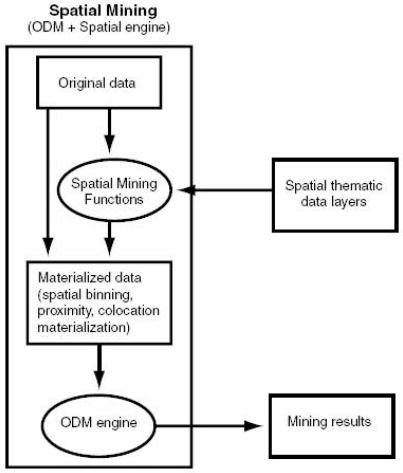


Abbildung 5.11: Oracle Mining

Eine besondere Aufmerksamkeit gebührt der Materialisierung der Autokorrelation (Nachbarschaftsbeziehungen), da diese beim Spatial Data Mining eine entscheidende Rolle einnimmt. Die Methoden im SAM-Package versuchen diesen Aspekt auf verschiedene Arten, 1. räumliche Diskretisierung (z. B. Kategorisierung nach Nord, Süd, Ost und West), 2. Materialisierung von Nachbarschaftsbeziehungen (z. B. Betrachtung eines Bereichs um einen Punkt oder Betrachtung der nächsten *n* Nachbarn), 3. Identifizierung von *co-located* Objekten (z. B. Videothek in der Nähe von FastFood-Restaurant) , zu beschreiben und letztendlich auf nicht-räumliche Attribute, die den räumlichen Einfluss weitestgehend beinhalten, abzubilden.

Das SAM-Package in Oracle unterstützt mit den bereitgestellten Methoden folgende Punkte, die entweder zur reinen (manuellen) Analyse der räumlichen Daten oder zum Spatial Data Mining durch die Materialisierung von räumlichen Beziehungen für eine Menge von thematischen Karten genutzt werden können.

- Cluster-Analyse, die geometrische Objekte werden nur anhand von Ortskoordinaten und unter Verwendung eines R-Baum Indexes in *n* Cluster unterteilt, wobei *n* beliebig gewählt werden kann. Die Cluster werden mittels der *Minimum-Bounding-Box*, die die geometrischen Objekte umschließt, dargestellt.
- Materialisierung der räumlichen Autokorrelation zu nicht-räumlichen Attributen durch Aggregierung der Werte der benachbarten Objekte, z. B. Betrachtung der Bevölkerungsdichte ei-

nes Objektes und der Bevölkerungsdichte im Umkreis von 5 km oder der Bevölkerungsdichte der benachbarten geometrischen Objekten.

- Co-Location Mining, bei der zwei thematische Karten mit ihren geometrischen Spezifikationen übereinander gelegt werden und anschließend die Co-Locations der geometrischen Objekte ermittelt werden, z. B. Welche Städte mit einer Bevölkerung von mehr als 100000 Einwohnern liegen an einer Autobahn?
- Spatial Diskretisierung, bei der eine Diskretisierung in Form einer Aufteilung der Karte, z. B. Nord, Süd, Ost, West, vorliegt und anschließend die geometrischen Objekte durch Berechnung der Schnittmenge von Objekt und Aufteilung der Karte einem diskretisierten Wert zugeordnet werden.

In [Oraa] wird die Anwendung und Funktionsweisen der Methoden von Spatial Analysis und Mining anschaulich an Beispielen zur Kriminalitätsrate einer Region beschrieben. Das Spatial Data Mining unter Anwendung der Methoden im SAM-Package wird anhand einer Klassifikation der Kriminalitätsrate in vier Bereiche (*gering, mittel, hoch, extrem hoch*) demonstriert. Dazu wird die Tabelle, welche die notwendigen Daten für die Klassifikation enthält, um zwei weitere Attribute erweitert, die den nachbarschaftlichen Einfluss zum einen begrenzt auf 500 m und zum anderen begrenzt auf die fünf nächsten Nachbarn repräsentieren. Das Ergebnis nach [Oraa] beschreibt eine Genauigkeitssteigerung der Klassifikation von $\approx$ 50% (62% ohne und 92% mit Betrachtung der räumlichen Autokorrelation).

5.2.4 Zusammenfassung

In den vorangegangenen Abschnitten wurden unterschiedliche Herangehensweisen zum Spatial Data Mining in räumlichen Datenbanken vorgeschlagen. Es wurden dabei Methoden und Algorithmen vorgestellt, die die Herausforderungen des Data Mining in räumlichen Datenbanken, d. h. komplexe Datenstrukturen, implizite Beziehungen und die Autokorrelation, zu lösen versuchen und somit ein Spatial Data Mining zu ermöglichen. Die Herangehensweisen lassen sich unabhängig von der Art der Problemstellung (d. h. Clustering, Assoziationsanalyse, etc.) in zwei Strategien unterteilen:

1. Materialisierung der räumlichen Daten und impliziten Beziehungen
2. Modifikation (Anpassung) der Data Mining-Methoden

In Abb. 5.12 sind die Strategien des Data Mining in räumlichen Datenbanken anschaulich zusammengefasst.

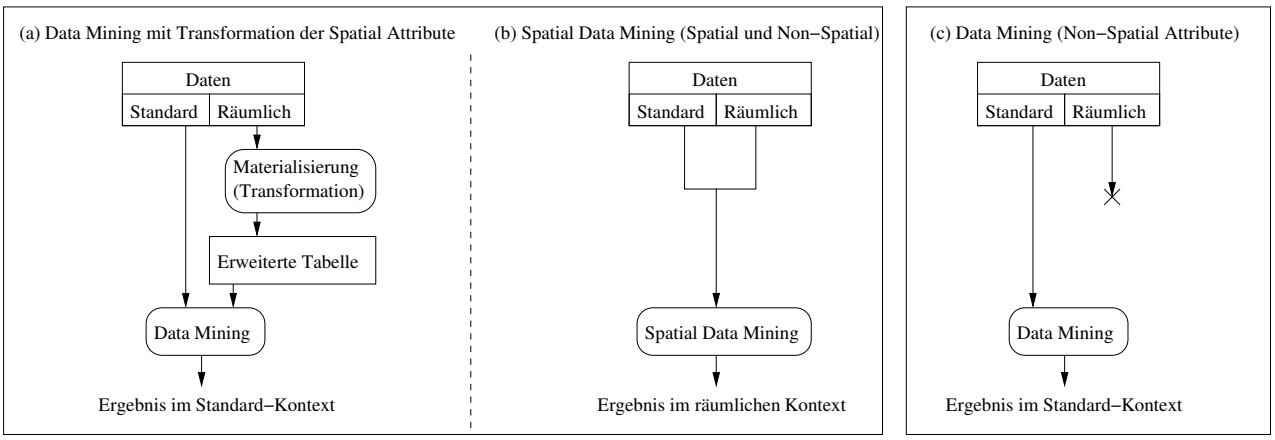


Abbildung 5.12: Vorgehensweisen beim Spatial Data Mining

Die Materialisierung der räumlichen Daten und impliziten Beziehungen (Abb. 5.12 (a)) beschränkt sich überwiegend auf die Transformation von topologischen und metrischen Beziehungen in Prädikate (z. B. `schneidet(X, 'Fluss') = true` oder `in_der_Nähe_von(X, 'Meer')`), die dann wiederum in Attribute umgewandelt werden. Die gerichteten Beziehungen (z. B. `X NordWest_von Y`) werden hingegen kaum bei der Materialisierung berücksichtigt [VB06b]. Ebenso wird keine Materialisierung von geometrischen Objekten (z. B. Polygon, Linie) in Standard-Datentypen durchgeführt (Ursache liegt wahrscheinlich darin, dass lediglich Clustering-Methoden von einer Ortsangabe/Lage der geometrischen Objekten Gebrauch machen).

Weitere Materialisierungen befassen sich mit Abbildung der Autokorrelation durch Aggregierung von numerischen Attributen der Nachbar-Objekte oder der Nachbarschaft eines Objektes [Ora06g]. Überdies beschäftigen sich andere Materialisierungen mit der Definition von räumlich motivierten Datensätzen (z. B. räumliche Transaktionen [RT05] für die Assoziationsanalyse), durch die Auffassung der räumlichen Datenbank als eine Überlagerung von thematischen Karten und der anschließenden Zusammenführung der nicht-räumlichen Attribute zu Datensätzen.

Die Modifikation der Data Mining-Methoden für das Data Mining in räumlichen Datenbanken (Abb. 5.12 (b)) umfasst ein breites Spektrum. Ein Vorgehen bei der Anpassung besteht in der Neudefinition von Begrifflichkeiten in den Methoden, z. B. Support [HPX06], Dichte [SEKX98] und Ähnlichkeit [RT04]. Ein weiteres Vorgehen besteht in der Integration von Nachbarschaftsbeziehungen in Form von Nachbarschaftsgraphen [EKS97] [Mor01]. Die Nachbarschaftsgraphen werden dann von den Methoden zum Spatial Data Mining und der Verarbeitung von räumlichen Daten und impliziten Beziehungen verwendet. Ein letztes Vorgehen, das in diesem Abschnitt vorgestellt wurde, besteht in der Anpassung des statistischen Modells [SC03], wodurch primär der Aspekt der Integration der Autokorrelation in das Data Mining-Verfahren verfolgt wird.

Anmerkung: Klassisches Data Mining in räumlichen Datenbanken

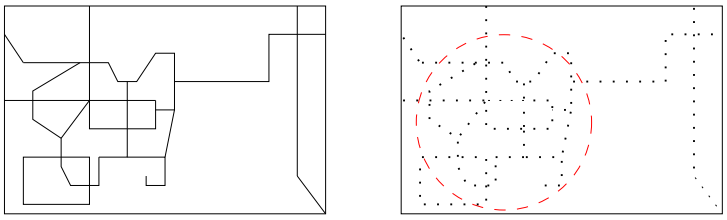
> Data Mining kann natürlich auch ohne Materialisierung der räumlichen Daten und impliziten Beziehungen mit den klassischen Methoden (Abb. 5.12 (c)) durchgeführt werden, wobei lediglich nicht-räumliche Attribute (d. h. numerische und nicht-numerische Standard-Datentypen) verarbeitet werden. Diese Art des Data Mining entspricht dem Vorgehen nach Kapitel 2. In [SC03] wird dieses Vorgehen des Data Mining in räumlichen Datenbanken, also lediglich auf nicht-räumlichen Attributen, jedoch ausgeschlossen, da in keiner Weise ein räumlicher Aspekt involviert ist und die eigentliche Motivation, d. h. die Verarbeitung von räumlichen und nicht-räumlichen Daten, nicht erreicht wird.

5.3 Materialisierung von geometrischen Objekten und räumlichen Beziehungen

In diesem Abschnitt wird ein Konzept vorgeschlagen, mit dem räumliche Daten und implizite räumliche Beziehungen durch eine geeignete Transformation in Standard-Datentypen umgewandelt werden können, so dass eine Knowledge Discovery mit Spatial Data Mining unter Anwendung von klassischen Methoden möglich ist. Dabei wird versucht für die einzelnen Data Mining-Methoden zweckmäßige und sinnvolle Transformationen zu entwickeln, um möglichst viele Informationen zu konservieren.

Die Motivation dieser Materialisierung besteht in der Wiederverwendung der Data Mining-Methoden, die durch die Oracle Data Mining-Cartridge (siehe Kapitel 3) bereitgestellt werden. Eine Erweiterung oder Modifikation der Methoden für das Spatial Data Mining ist in Oracle Data Mining nicht möglich.

5.3.1 Beispiel für die Materialisierung von räumlichen Informationen



(a) Liniendarstellung (b) Punktdarstellung

Abbildung 5.13: Transformation für Spatial Clustering

In Abb. 5.13 (a) ist ein Ausschnitt einer Karte dargestellt, in der Straßen abgebildet sind. Die Data Mining-Aufgabe besteht darin, `Siedlungen` anhand der Straßen zu identifizieren. Es wird vermutet, dass eine Ansammlung von Straßenzügen ein `Siedlungsgebiet` impliziert. Für die Lösung des Problems soll eine Clustering-Methode angewendet werden, um Ballungsgebiete und somit `Siedlungen` zu identifizieren.

Bei dieser Problemstellung ergeben sich für das Clustering mit einer klassische Methode folgende Schwierigkeiten:

- **Repräsentation der Straßen** – Straßen werden in räumlichen Datenbanken durch Linien bzw. Linienzüge dargestellt, wodurch ein Clustering ohne Transformation der komplexen Datenstruktur nicht möglich ist.
- **Transformation der Straßen** – Für die komplexe Datenstruktur der Linie bzw. des Linienzüges wird keine Transformation durch das Datenbanksystem bereitgestellt.

Durch eine adäquate Transformation der Straßen – allgemein geometrischer Objekte – in eine Punktdarstellung (Abb. 5.13 (b)) könnten diese von den klassischen Clustering-Methoden verarbeitet werden. Der Cluster in Abb. 5.13 (b) – roter Kreis – könnte auf diese Weise identifiziert werden.

Die Materialisierung der räumlichen Daten (d. h. Autokorrelation) und der impliziten räumlichen Beziehungen ergeben sich aus den vorgestellten Methoden in der Literatur.

Folgende Lösungsvorschläge und Methoden aus der Literatur werden für das Konzept zur Materialisierung von räumlichen Informationen adaptiert:

- Oracle Spatial Analysis and Mining (SAM-Package) [Ora06g] für die Materialisierung von Nachbarschaftsbeziehungen und der Integration der Autokorrelation von numerischen Attributen
- Schichten-Modell nach [RT05] für die Materialisierung von impliziten topologischen, metrischen und gerichteten Beziehungen
- Materialisierung von topologischen und metrischen Beziehungen nach dem Prinzip *feature type* und *feature instance* [VB06b][VB05b]
- Materialisierung von räumlichen Daten nach dem *reference-feature centric* und *data partitioning*-Prinzip[SC03] und der Überlagerung nach dem Schichten-Modell [RT05]

5.3.2 Was wird für das Spatial Data Mining materialisiert?

Im Folgenden sind die Punkte aufgelistet, für die eine Materialisierung durchgeführt werden soll:

1. Geometrische Objekte – Diese beschreiben in den meisten Fällen die Form der räumlichen Informationen, z. B. Polygon beschreibt ein Waldgebiet oder eine Linie beschreibt eine Straße. Um die Informationen, die in den geometrischen Objekten enthalten sind, beizuhalten, müssen diese in eine geeignete Repräsentation durch Standard-Datentypen überführt werden.
2. Allgemeine Nachbarschaftsbeziehungen bezogen auf nicht-räumliche Attribute – Diese beschreiben den Aspekt der Autokorrelation im Bezug auf nicht-räumliche Attribute, d. h. die gegenseitige nachbarschaftliche Beeinflussung von räumlichen Objekten. Um den Einfluss der

Autokorrelation bei der Transformation beibehalten zu können, müssen die nicht-räumlichen Informationen der benachbarten Objekte im betrachteten Objekt abgebildet werden.

3. Topologische Beziehungen – Diese beschreiben, wie bereits in Abschnitt 5.1 erläutert, Beziehungen zwischen geometrischen Objekten. Um diese Informationen vollständig beizubehalten, sind so viele Transformation notwendig wie es topologische Beziehungen zwischen den geometrischen Objekten gibt.
4. Metrische Beziehungen – Diese beschreiben, wie bereits in Abschnitt 5.1 erläutert, Beziehungen zwischen geometrischen Objekten, die die Metrik betreffen, z. B. Abstand zweier Objekte oder Fläche eines Objektes. Diese Informationen lassen sich durch Berechnungen abbilden, z. B. Abbildung des euklidischen Abstands eines `Wohngebiets` zum `Meer` auf einen numerischen Standard-Datentyp.
5. Gerichtete Beziehungen – Diese Art von Beziehungen werden gewöhnlich beim Spatial Data Mining vernachlässigt [VB06b], werden jedoch aus Gründen der Vollständigkeit ebenfalls materialisiert. Dabei erfolgt eine Materialisierung einer gerichteten Beziehung durch die Abbildung der relativen Lage zwischen zwei geometrischen Objekten.

Anmerkung: Das allgemeine Vorgehen der Materialisierung besteht in der Erweiterung einer einzigen Objekt-Tabelle um Attribute, die die materialisierten räumlichen Informationen beinhalten und für die Methoden und Algorithmen des klassischen Data Mining – Oracle Data Mining – bereitgestellt werden.

5.3.3 Transformation geometrischer Objekte

Bei der Transformation von geometrischen Objekten stellt sich zunächst die Frage nach der Betrachtungsweise des Problems, d. h. welches Ziel soll mit der Transformation erreicht werden? In diesem Zusammenhang ergeben sich die folgenden zwei Betrachtungen:

1. Geometrische Betrachtung, z. B. Informationen über die Beziehung der geometrischen Objekte X, Y mit X `benachbart_zu` Y oder X `schneidet` Y
2. Thematische Betrachtung, z. B. die Identifizierung von Regionen mit erhöhter Brandgefahr oder Kriminalitätsrate

Bei der geometrischen Betrachtung müssen, bei der Transformation der geometrischer Objekte, die Beziehungen zwischen den Objekten abgespeichert werden. Die Speicherung dieser Informationen (z. B. X `schneidet` Y) kann entweder durch eine externe Tabelle erfolgen, die die einzelnen Beziehungen speichert, oder durch die Erweiterung der Tabelle um eine Menge von Attributen, wobei sich die Anzahl der Attribute nach der Anzahl der geometrischen Objekte richtet.

Jedoch sind beide Lösungen zur Transformation von geometrischen Objekten nach der geometrischen Betrachtungsweise unzureichend, da zum einen die Data Mining-Methoden jeweils nur eine einzelne Objekt-Tabelle verarbeiten und deshalb die externe Tabelle mit den gespeicherten Beziehungen nicht berücksichtigen können. Und zum anderen führt die Erweiterung der Tabelle um eine Menge Attribute zu großen Tabellen, die evtl. nicht mehr verarbeitet werden können.

Bei der thematischen Betrachtung sind die geometrischen Beziehungen zwischen Objekten untergeordnet. Die relevanten Information sind die thematischen Attribute bzgl. dieser Objekte. Demnach ist lediglich entscheidend, dass durch die Transformation die geometrischen Objekte hinreichend genau repräsentiert werden. Daher wird im Folgenden eine Transformation vorgeschlagen, die eine thematischen Betrachtungsweise der geometrischen Objekte verfolgt.

Die Transformation von beliebigen geometrischen Objekte (z. B. Polygon, Linien, etc.) soll durch eine Abbildung der Geometrie auf einen oder mehrere repräsentative Punkte P_i, d. h. $P_i = (x, y)$

durchgeführt werden. Durch die reduzierte Repräsentation der geometrischen Objekte durch Punkte werden jedoch die topologischen Beziehungen komplett vernachlässigt und die metrischen Beziehungen verfälscht. In Abb. 5.14 ist dieser Zusammenhang anschaulich dargestellt. Die Polygone A und C besitzen die topologische Beziehung *benachbart* und die Polygone B und C *überlappt*. Durch die Repräsentation durch Punkte gehen diese Informationen vollständig verloren.

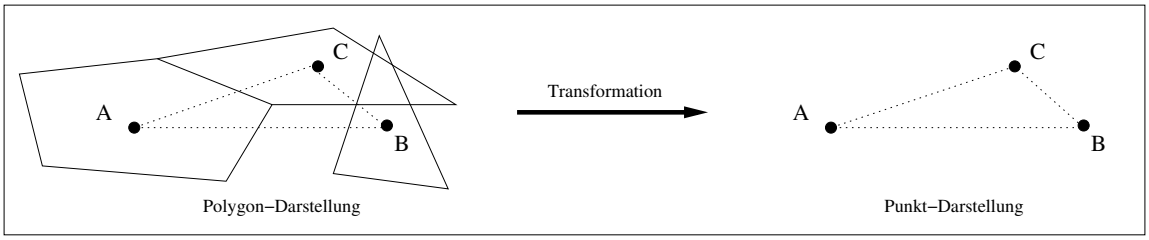


Abbildung 5.14: Verlust der topologischen Beziehungen zw. den Geometrien

Die Verfälschung der metrischen Beziehungen schildert folgendes Beispiel. Z. B. sind die Polygone A und C benachbart und haben den Abstand 0 voneinander. Durch die Punkt-Darstellung von A und C ist der Abstand jedoch > 0.

Um diese Unzulänglichkeiten bei der Konservierung der metrischen Beziehungen zu minimieren, müssen entsprechend Transformationen für die geometrischen Objekte definiert werden. Im Folgenden wird durch die thematische Betrachtungsweise der Aspekt gebilligt, dass die topologischen Beziehungen zwischen den geometrischen Objekten durch die Repräsentation durch Punkte komplett verloren gehen.

Im Weiteren werden für die Transformation lediglich die Geometrien betrachtet, die in [ORAb] und [Tie03] beschrieben sind. Es ergeben sich daraus folgende Geometrie-Transformation, die in Abb. 5.15 anschaulich dargestellt sind. *Anmerkung:* Andere geometrische Objekte, z. B. Kreisbogen String, *Collections*, werden ignoriert.

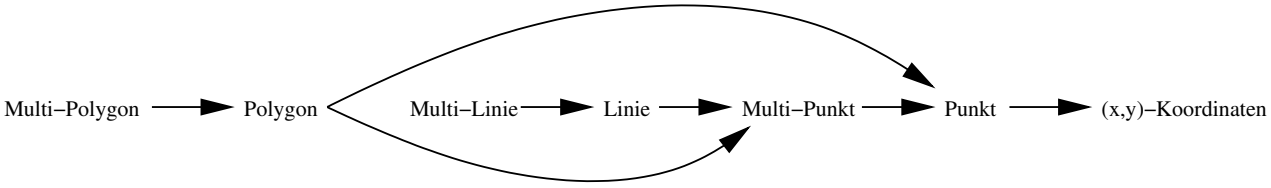


Abbildung 5.15: Transformation der geometrischen Objekte zu Punkten-Koordinaten

Die Transformation von Punkten ist trivial. Es werden lediglich die Koordinaten des Punktes auf zwei neue Attribute abgebildet. In Abb. 5.16 ist diese Transformation zu sehen, in der die Punkt-Geometrie in X und Y aufgeteilt wird. Die nicht-räumlichen Attribute bleiben bei der Transformation unverändert.

ID	Geometrie	Attribute
1	Punkt	...

→

ID	Koordinate X	Koordinate Y	Attribute
1	Punkt.X	Punkt.Y	...

Abbildung 5.16: Transformation von Punkten

Die Transformation der Geometrie Multi-Punkt wird durch mehrfache Anwendung der Transformation der Punkt-Geometrie durchgeführt. Durch die Abbildung wird aus einem Geometrie-Objekt (Multi-Punkt) mehrere Geometrie-Objekte (Punkte). Die nicht-räumlichen Attribute bleiben bei der Transformation unverändert und werden einfach von den Punkt-Darstellungen übernommen. In

Abb. 5.17 ist diese Transformation dargestellt, wobei die Anzahl der Einträge in der rechten Tabelle abhängig von der Anzahl der Punkte im Multi-Punkt-Objekt ist.

ID	Geometrie	Attribute
1	Multi−Punkt	...

→

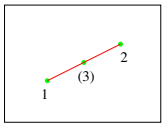
ID	Koordinate X	Koordinate Y	Attribute
1.1	Punkt.X	Punkt.Y	...
1.2	Punkt.X	Punkt.Y	...
1.3	Punkt.X	Punkt.Y	...

Abbildung 5.17: Transformation einer Menge von Punkten (Multi-Punkt)

Die Transformation der Geometrie Linie wird durch eine Abbildung der Endpunkte als Repräsentanten der Linie durchgeführt. Da die Reduzierung der Linie auf die Endpunkte bei längeren Linien zu unpräzise ist, ist daher ein Parameter einzuführen, der bestimmt, wann die Linie durch Zwischenpunkte angereichert werden soll. In Abb. 5.18 ist die Transformation einer Linie dargestellt, wobei der Punkt 3 einen zusätzlichen Punkt repräsentiert, der eingefügt wird, wenn die Linie zu *lang* ist. Die nicht-räumlichen Attribute bleiben bei der Transformation unverändert und werden einfach von den Punkt-Darstellungen übernommen.

Einfache Linie

ID	Geometrie	Attribute
1	Linie	...

→ 

→

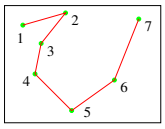
ID	Koordinate X	Koordinate Y	Attribute
1	Linien−Punkt_1.X	Linien−Punkt_1.Y	...
2	Linien−Punkt_2.X	Linien−Punkt_2.Y	...
(3)	(Linien−Punkt_3.X)	(Linien−Punkt_3.Y)	(...)

Abbildung 5.18: Transformation einer Linie zu einer Menge von Punkten

Die Transformation der Geometrie Multi-Linie wird durch mehrfache Anwendung der Transformation der Linien-Geometrie durchgeführt, d. h. die Multi-Linie wird in ihre Linien-Segmente zerlegt und anschließend nach der obigen Transformation einer einfachen Linie transformiert. In Abb. 5.19 ist diese Transformation dargestellt. Die nicht-räumlichen Attribute bleiben bei der Transformation wiederum unverändert und werden einfach von den Punkt-Darstellungen übernommen.

Zusammengesetzte Linie

ID	Geometrie	Attribute
1	Multi−Linie	...

→ 

→

ID	Koordinate X	Koordinate Y	Attribute
1.1	Linien−Punkt_1.X	Linien−Punkt_1.Y	...
1.2	Linien−Punkt_2.X	Linien−Punkt_2.Y	...
...	...	...	...
...	...	...	...
1.n	Linien−Punkt_n.X	Linien−Punkt_n.Y	...

Abbildung 5.19: Transformation einer Multi-Linie zu einer Menge von Punkten (Multi-Punkt)

Im Folgenden wird ein Algorithmus angegeben, der die Zerlegung von Geometrien vom Typ Linie und Multi-Linie durchführt.

```
EINGABE: 1. Geometrie: Linie oder Multi-Linie
         2. Wert für max. Länge der Linie
AUSGABE: relationale Tabelle mit materialisierten Geometrien
METHODE:
       1. WENN Geometrie vom Typ Multi-Linie ODER
              Länge der Linie > max. Länge  DANN
          1.1 Zerlege Linie in kleinere Linien-Segmente
          1.2 Rekursiver Aufruf dieser Methode mit den Linien-Segmenten
```

```
2. SONST
      2.1 Extrahiere die Koordinaten X und Y
          von Start- und Endpunkt der Linie
      2.1 Füge die X, Y - Paare von Start- und Endpunkt
          und nicht-räumliche Attribute in die Tabelle ein
```

Die Transformation von Polygon-Geometrien gestaltet sich ein wenig komplizierter, da sie nach [ORAb] in verschiedenen Formen auftreten, z. B. konvexe und nicht-konvexe Polygone, Polygon mit Aussparungen, optimierte Polygone.

Um alle diese Polygone auf die gleiche Weise verarbeiten zu können, wird eine Abstraktion vorgenommen, wobei jeweils nur die konvexe Hülle der verschiedenen Polygone betrachtet wird. (Dadurch werden die Sonderfälle bei der Transformation von Polygonen minimiert.) Durch diese vereinfachte Betrachtung sind nur konvexe Polygone zu transformieren. Sie werden dann durch den Centroiden (Schwerpunkt des Polygons) repräsentiert. In Abb. 5.20 (oben) wird ist die Transformation dargestellt, wobei die nicht-räumlichen Attribute einfach übernommen werden.

Es weiterer Aspekt darin, dass Polygone flächenmäßig sehr groß im Vergleich zur Größe des Bezugssystems sein können. Durch die Repräsentation der Polygone durch Punkt-Geometrie entsteht dadurch jedoch eine nicht zu verachtende Ungenauigkeit (metrische Beziehungen werden verzerrt). Um diese zu minimieren werden diese Polygone in kleine Polygone unterteilt. Eine einfache Unterteilung stellt ein Raster dar, das über das Polygon gelegt wird. Die neuen Polygone, die im Vergleich kleiner als das Ursprungspolygon sind, werden anschließend jeweils durch ihren Centroiden repräsentiert. Die Zerlegung des Polygons und die Transformation in ein oder mehrere repräsentative Punkte ist in Abb. 5.20 (unten) dargestellt. Die Handhabung der nicht-räumlichen Attribute wird weiter hinten im Abschnitt behandelt, da diese zum Teil nicht ohne weiteres unverändert übernommen werden können.

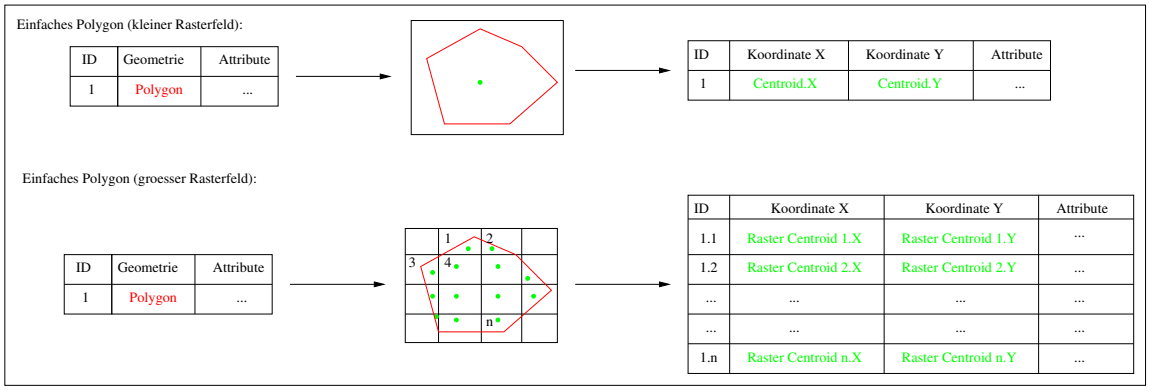


Abbildung 5.20: Unterteilung *großer* Polygone und Darstellung durch eine Menge von Punkten

Die Transformation von Multi-Polygonen, d. h. einer Menge von Polygonen, wird durch mehrfache Anwendung der Transformation von Polygonen durchgeführt. Bei der Zerlegung des Multi-Polygon-Objektes in die einzelnen Polygone müssen die nicht-räumlichen Attribute differenziert behandelt werden. Diese Behandlung orientiert sich am Vorgehen bei der Transformation von Polygonen, die im Folgenden beschrieben wird.

Anmerkung:

Bei der Unterteilung der Polygone oder Multi-Polygone bleibt die jeweilige Zuordnung der kleineren Polygone zu ihrem Ursprungspolygon erhalten.

Probleme bei der Transformation von Polygonen:

Bei der vorgeschlagenen Transformation der Polygone können jedoch Probleme auftreten, die durch die Gestalt der Geometrien verursacht werden. In Abb. 5.21 sind einige Beispiele dargestellt. Durch die Abstraktion der Geometrie durch die konvexe Hülle können Geometrien mit sich stark unterscheidenden Eigenschaften entstehen. Besonders deutlich macht sich dabei die Veränderung der Fläche einer Geometrie bemerkbar. Ist ein auffallend großer Unterschied zwischen der Fläche des eigentlichen Polygons und des Polygons, das durch die konvexe Hülle repräsentiert wird, zu verzeichnen, so sollte das Polygon nicht durch die konvexe Hülle dargestellt werden. In diesem Fall ist es plausibel, dass das Polygon ebenfalls durch ein Raster unterteilt wird, um somit eine genauere Repräsentation durch eine Punktmenge zu erreichen.

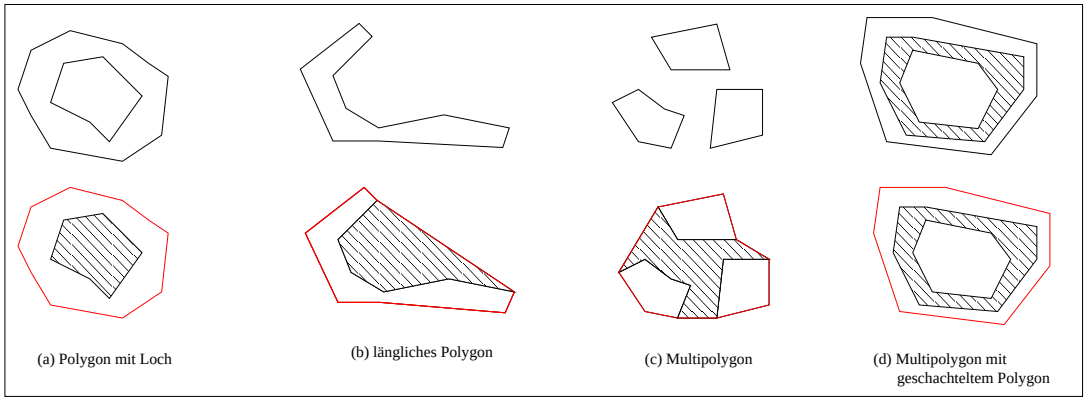


Abbildung 5.21: Probleme bei der Transformation von Polygonen

Da es bei diesem Vorgehen jedoch vorkommen kann, dass ein Polygon bereits sehr klein ist, aber dennoch eine große Diskrepanz zwischen der Fläche des Polygons und der Fläche konvexen Hülle aufweist, bietet sich hierbei ein Richtwert an, der angibt, wann ein Polygon nicht mehr zerlegt werden darf.

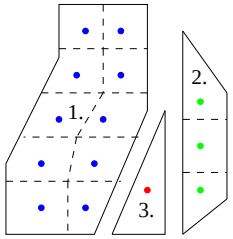
Im Folgenden wird ein Algorithmus angegeben, der die Zerlegung von beliebigen Polygonen, z. B. Polygone mit Löchern, abstrahiert durch die konvexe Hülle realisiert:

```
EINGABE: 1. Geometrie: Polygon
         2. Werte für max. Flächenunterschied und min. Fläche zur Zerlegung
AUSGABE: relationale Tabelle mit materialisierten Geometrien
METHODE:
      1. Berechne die konvexe Hülle
      2. WENN Fläche Polygon < min. Fläche zur Zerlegung DANN
          2.1 Gehe zu Punkt 4.
      3. WENN Fläche der konvexen Hülle >> Fläche Polygon DANN
          3.1 Zerlege Polygon anhand eines Rasters in kleine Polygone
          3.2 Berechne nicht-räumliche Attribute für die neuen Polyone
          3.3 Rekursiver Aufruf dieser Methode mit den neuen Polygonen
      4. SONST
          4.1 Berechne den Centroiden des Polygons und Extrahiere X und Y
          4.2 Füge X, Y und die nicht-räumlichen Attribute in die Tabelle ein
```

Behandlung von nicht-räumlichen Attribute:

Bei der Zerlegung von Polygonen und Multi-Polygonen ist es notwendig zu entscheiden, was mit den nicht-räumlichen Attributen, die das Polygon beschreiben, passieren soll. Dieser Aspekt soll anhand eines kurzen Beispiels erläutert werden.

In Abb. 5.22 ist eine Karte mit drei Waldflächen gegeben, die durch die Attribute Art des Waldgebiets (`Art des Waldes`), Größe des Waldgebiets (`Fläche`) und Anzahl der Brände für das Waldgebiet (`Anzahl an Bränden`) als Maß für die Brandgefahr beschrieben sind.



ID	Art der Waldes	Fläche (qkm)	Anzahl an Bränden	Unterteilung	Anzahl an Bränden (Unterteilung)
1.	Laub	100	80	10	8
2.	Misch	30	60	3	20
3.	Nadel	10	25	1	25

Abbildung 5.22: Beispiel: Behandlung von nicht-räumlichen Attribute

In diesem Beispiel soll eine Identifikation von gefährdeten Regionen (d. h. viele Brände) durch die Betrachtung des Attributs `Anzahl an Bränden` ermittelt werden. Bei der Zerlegung der Waldgebiete in repräsentative Punkte ergeben sich zwei Alternativen zur Behandlung des Attributs `Anzahl an Bränden`. Zum einen wird das Attribut unverändert übernommen, was der Interpretation des gesamten Waldgebiets entspricht, und zum anderen wird das Attribut prozentual, bezogen auf die Fläche, auf die Repräsentanten aufgeteilt.

Im ersten Fall wird die Region 1 und im zweiten Fall die Regionen 2 und 3 als gefährdet identifiziert. Ausgehend von der gegebenen Problematik der Identifikation von gefährdeten Region ergibt sich, dass die prozentuale Aufteilung des Attributs `Anzahl an Bränden` auf die Repräsentanten eine aussagekräftigere Interpretation für das Problem liefert. Dies ist dadurch zu begründen, dass die Region 1 ein flächenmäßig großes Gebiet darstellt, für welches die nicht-räumlichen Attribute gelten. Dadurch ergibt sich jedoch eine ungenaue Darstellung der Situation, wenn die Region in kleinere Regionen unterteilt wird.

Bei der Unterteilung von geometrischen Objekte in mehrere Teile, liegt es daher nahe, dass auch die nicht-räumlichen Attribute aufgeteilt werden. Dabei sollte zwischen den numerisch-aufteilbaren, numerisch-nicht-aufteilbaren und nicht-numerischen Attributen unterschieden werden. Die numerisch-nicht-aufteilbaren Attribute (z. B. eine Altersangabe, Jahreszahl) und die nicht-numerischen Attribute (z. B. Bezeichnung, Typ, etc.) bleiben unverändert und werden für die neuen Polygone übernommen.

Für die Behandlung von numerisch-aufteilbaren Attributen ergeben sich zwei Vorgehensweisen, die im Folgenden aufgelistet sind:

1. unveränderte Übernahme der numerischen Attribute
2. prozentuale Aufteilung von numerischen Attribute

Die erste Variante ist einfach zu realisieren, hat jedoch den entscheidenden Nachteil, dass Polygone, die in kleinere Polygone aufgeteilt worden sind, eine Verzerrung und Verfälschung der Datenbasis darstellen können. Es sei beispielsweise ein Polygon in vier kleinere Polygone unterteilt, dann besitzen alle Polygone die gleichen nicht-räumlichen Eigenschaften. Würde anschließend die Dichte für die einzelnen Polygone, z. B. Bevölkerungsdichte oder Kriminalitätsrate, berechnet werden, dann würden sich aufgrund der unterschiedlichen Größen verschiedene Dichten ergeben.

Die zweite Variante zur Behandlung von numerisch-aufteilbaren Attributen ist semantisch motiviert. Das Vorgehen bei der Aufteilung von nicht-räumlichen Attributen ist daher situationsabhängig. Die Polygone werden bei dieser Variante nach ihrem prozentualen Anteil am Ursprungspolygon mit den numerischen Attributen versehen, z. B. die Überdeckung der Polygone beträgt 20%, dann ist der Wert für ein numerisches Attribut ebenfalls 20%.

Im Folgenden wird ein Algorithmus zur Transformation von geometrischen Objekten angegeben, wobei Punkte, Multi-Punkte, Polygone und Multi-Polygone transformiert werden:

```
EINGABE: 1. Tabelle mit nicht-räumlichen Attributen, einem räumlichen Attribut
         2. Größe A, bei der ein Polygon unterteilt wird
AUSGABE: relationale Tabelle mit materialisierten Geometrien
METHODE:
      1. Erstelle eine Tabelle (Rückgabe der Methode)
         (Spalten der Tabelle ergeben sich aus nicht-räumlichen
          Spalten der Eingabe-Tabelle und den neuen Spalten X und Y)
      2. Für jede Geometrie in der Tabelle:
      3. Bestimme die Art der Geometrie
      4. WENN Geometrie vom Typ Punkt DANN
          4.1 Extrahiere die Koordinaten X und Y.
          4.2 Füge X, Y und nicht-räumliche Attribute in die Tabelle ein
          4.3 Gehe zu Punkt 2.
      5. WENN Geometrie vom Typ Multi-Punkt DANN
          5.1 Zerlege Multi-Punkt in mehrere Punkte
          5.2.Für jeden Punkt extrahiere die Koordinaten X und Y.
              5.2.1 Füge X, Y und nicht-räumliche Attribute in die Tabelle ein
          5.3 Gehe zu Punkt 2.
      6. WENN Geometrie vom Typ Linie oder Multi-Linie DANN
          6.1 Behandle Geometrie nach der obigen Methode für Linien
          6.2 Gehe zu Punkt 2.
      7. WENN Geometrie vom Typ Polygon oder Multi-Polygon DANN
          7.1 Behandle Geometrie nach der obigen Methode für Polygone
          7.2 Gehe zu Punkt 2.
      8. WENN Geometrie nicht vom unterstützten Typ DANN
          8.1 Ignoriere Datensatz und gehe zu Punkt 2.
```

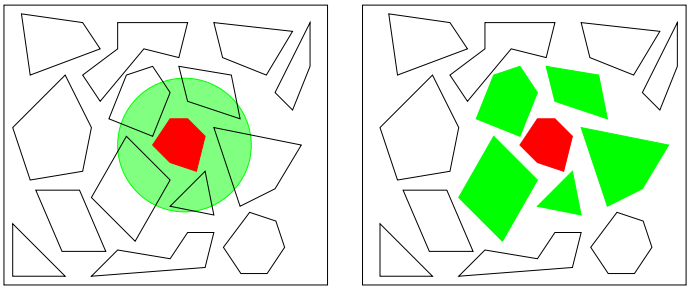
Bemerkungen zum Algorithmus:

In Oracle werden Methoden bereitgestellt, die Basis-Operationen auf geometrischen Objekten durchführen. Folgende Methoden, die im `SAM`-Package und `SDO_GEOM`-Package enthalten sind, könnten bei einer Implementierung benutzt werden:

- `SDO_AREA` – Berechnet die Fläche eines Polygons
- `SDO_CONVEXHULL` – Berechnet die konvexe Hülle eines Polygons
- `SDO_CENTROID` – Berechnet den Centroiden einer Geometrie
- `AGGREGATES_FOR_GEOMETRY` – Berechnet den prozentualen Anteil einer Geometry zu einer anderen Geometrie
- `AGGREGATES_FOR_LAYER` – Berechnet den jeweiligen prozentualen Anteil aller Geometrien in Bezug auf eine Menge anderer Geometrien, z. B. Raster über eine Geometrie (Polygon) gelegt
- `TILED_BINS` – Erzeugt ein Raster nach vorgegebener Granularität

Durch die Materialisierung der räumlichen Attribute hat die Tabelle nun ausschließlich Standard-Datentypen und kann mit den Mechanismen der Oracle Data Mining-Cartridge verarbeitet werden.

5.3.4 Materialisierung von Nachbarschaftsbeziehungen



(a) Bereich um Objekt X (b) 5 − Nächsten Nachbarn

Abbildung 5.23: Beschränkung des Einflusses auf eine festgelegte Nachbarschaft

Das erste Gesetz der Geographie, also jedes Objekt wird durch seine Nachbar-Objekte beeinflusst, wobei der Einfluss mit zunehmender Entfernung abnimmt, soll in Form eines Attributes materialisiert werden. Dadurch soll der Aspekt der Autokorrelation beim Data Mining in räumlichen Datenbanken berücksichtigt werden. Die Integration dieser Eigenschaft sollte sich dabei lediglich auf die Betrachtung der nicht-räumlichen Attribute der Nachbar-Objekte beschränken. Außerdem sollte der nachbarschaftliche Einfluss begrenzt werden können, z. B. sollen nur die n nächsten Nachbarn oder die Nachbarn in einer maximalen Umgebung betrachtet werden. In Abb. 5.23 ist die Beschränkung der Einflussnahme graphisch veranschaulicht.

In Oracle werden im SAM-Package [ORAb] Methoden bereitgestellt, mit denen die nicht-räumlichen – beschränkt auf numerische – Attribute der Nachbar-Objekte betrachtet werden können. Dadurch ist eine Materialisierung der Autokorrelationseigenschaft von räumlichen Objekten möglich. Diese Methoden sind in der Anwendung recht flexibel, da die Einflussnahme der Nachbar-Objekte durch Angabe eines maximalen Bereichs oder einer festgelegten Anzahl an Nachbar-Objekten reguliert werden kann. Die Materialisierung kann mit den folgenden Methoden durchgeführt werden:

- `AGGREGATES_FOR_GEOMETRY` – Berechnet den prozentualen Anteil einer Geometry zu anderen Geometrien, d. h. sich überlappende Bereiche der Geometrien werden aggregiert. Der nachbarschaftliche Einfluss kann durch einen Radius um das Objekt, z. B. 2 km (Abb. 5.23 (a)), oder durch Angabe der n am nächsten benachbarten Objekten (Abb. 5.23 (b)) beschränkt werden.
- `AGGREGATES_FOR_LAYER` – Berechnet den jeweiligen prozentualen Anteil aller Geometrien einer Tabelle in Bezug auf eine Menge anderer Geometrien einer anderen Tabelle, d. h. die Methode führt die Methode `AGGREGATES_FOR_GEOMETRY` für jedes Geometrie-Objekt einzeln aus.

Die Materialisierung der Autokorrelation erfolgt durch das Hinzufügen von weiteren Attributen an die Objekt-Relation. Dabei wird für jedes spezifizierte (numerische) Attribut jeweils ein weiteres Attribut erzeugt. In Abb. 5.24 ist die Materialisierung der Autokorrelation schematisch dargestellt.

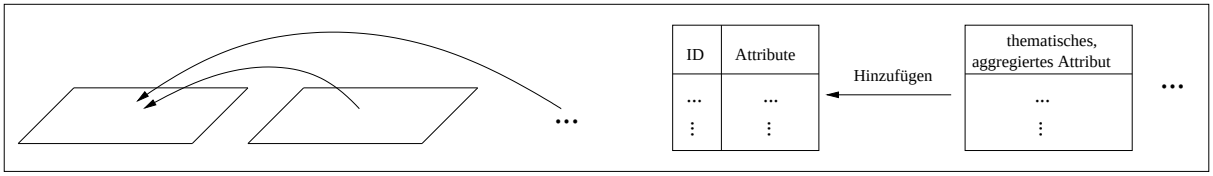


Abbildung 5.24: Materialisierung der Autokorrelationseigenschaft

Folgende Parameter für die Materialisierung sind für die Methoden im SAM-Package erforderlich:

- Eine Objekt-Relation (für die Methode `AGGREGATES_FOR_LAYER`), die ein Attribut mit Geometrie-Objekten besitzt, oder ein Geometrie-Objekt (für die Methode `AGGREGATES_FOR_GEOMETRY`).
- Eine thematische Karte, die das nicht-räumliche numerische Attribut vorgibt.
- Eine Aggregierungsfunktion (d. h. `AVG`,`MAX`, etc.), die die numerischen Werte der spezifizierten Nachbarschaft zusammenfasst.

- Spezifikation der Nachbarschaft (z. B. `'distance=5 unit=km'` entspricht einem Puffer von 5 km Radius um die Geometrie) oder der Anzahl der Nachbar-Objekte (z. B. `'sdo_num_res=5'` entspricht 5 nächsten Nachbar-Objekte)

Um die nicht-numerischen Attribute der Nachbar-Objekte ebenfalls abzubilden, ist ein anderer Mechanismus zu definieren. Eine Möglichkeit der Materialisierung der nicht-numerischen Attribute basiert auf dem Mehrheitsentscheid, d. h. bei Betrachtung von n Nachbar-Objekten wird der am häufigsten vorkommende Wert eines gegebenen Attributs als Repräsentant gewählt. Für die Realisierung bietet das `SDO_GEOM`-Package die Funktion `SDO_NN` an, die die n nächsten Nachbarn eines Objektes bestimmt. Im Folgenden wird ein Algorithmus zur Transformation der nicht-numerischen Attribute der n Nachbarn angegeben:

```
EINGABE: Geometrie-Objekt G, Attribut A und
         Anzahl der Nachbar-Objekte n
AUSGABE: Wert für das nicht-numerische Attribut
METHODE:
      1. Berechne die n nächsten Nachbar-Objekte von G
      2. Ermittle den häufigsten Wert für das Attribut A
      3. Rückgabe des ermittelten Wertes
```

5.3.5 Materialisierung von topologischen Beziehungen

Die Integration von materialisierten topologischen Beziehungen in das Spatial Data Mining ist besonders bei der Klassifikation und der Assoziationsanalyse überaus wichtig. Dadurch können räumliche Eigenschaften in die Klassifizierung von Objekten einbezogen werden, die beim Spatial Data Mining mit den klassischen Methoden ignoriert würden. Außerdem ist durch die Materialisierung eine Räumliche Assoziationsanalyse nach Abschnitt 5.2.1 und eine Co-Location Analyse durchführbar.

Beispiel: Für die Klassifizierung einer Region nach `Miete=hoch` und `Miete=niedrig` ist neben den Attributen des Objektes selbst ebenfalls die räumliche Umgebung relevant. Deshalb sollen in den Prozess der Klassifikation die folgenden zwei räumlichen Informationen integriert werden:

1. `In der Nähe von Objekt X befindet sich eine Autobahn.`
2. `In der Nähe von Objekt X befindet sich eine Parkanlage.`

Um diese Informationen in den Klassifikationsprozess mit dem klassischen Data Mining zu integrieren, müssen die topologischen Beziehungen durch eine Erweiterung der Objekt-Relation in Form von Attributen erfolgen. Im Beispiel ist eine Materialisierung der topologischen Beziehungen in die Attribute `Autobahn` und `Parkanlage` sinnvoll, wobei die Werte der Attribute entweder `TRUE`, d. h. topologische Beziehung ist gültig, und `FALSE`, d. h. topologische Beziehung ist **nicht** gültig, annehmen.

Die Materialisierung von topologischen Beziehungen wird nach dem *feature type*-Prinzip [VB06b] durchgeführt. Bei dieser Form der Materialisierung wird eine topologische Beziehung von einem hohen Abstraktionsgrad betrachtet. Das folgende Beispiel einer topologischen Beziehung

schneidet(Fluss='Leine', Stadt='Hannover') =TRUE

wird nach dem *feature type*-Prinzip folgendermaßen reduziert:

schneidet(Fluss, Stadt='Hannover') =TRUE

Mit der Materialisierung mit dem hohen Abstraktionsgrad soll lediglich ausdrückt werden, dass eine topologische Beziehung zwischen zwei geometrischen Objekten gültig ist. Dabei ist es weniger interessant, welche geometrischen Objekte daran beteiligt sind.

Technisch betrachtet wird die Materialisierung von topologischen Beziehungen durch das Hinzufügen von booleschen Attributen an die Objekt-Tabelle realisiert (Abb. 5.25). Dabei wird nach dem Schichten-Modell vorgegangen. Zunächst wird eine Referenz-Schicht (z. B. Städte), die Objekt-Tabelle, ausgewählt. Anschließend werden durch Überlagerung von thematischen Karten (z. B. Straßen oder Flüsse) mit der Referenz-Schicht die topologischen Beziehungen materialisiert.

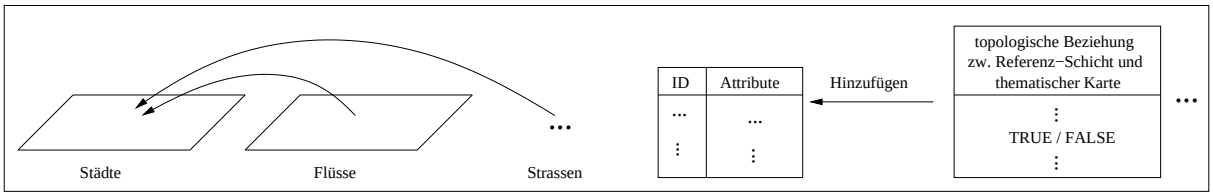


Abbildung 5.25: Materialisierung topologischer Beziehungen

Im Folgenden wird ein Algorithmus vorgestellt, der eine Transformation einer topologischen Beziehung in ein boolesches Attribut durchführt.

```
EINGABE: 1. Referenz-Schicht (Objekt-Tabelle)
         2. Schicht (Tabelle) - thematische Karte
         3. Art der topologischen Beziehung
AUSGABE: Referenz-Schicht erweitert um ein boolesches Attribut
METHODE:
      1. Erstelle eine neue Spalte in der Tabelle
         (Bezeichnung: topologische Beziehung und thematische Karte)
      2. Für jede Geometrie R in der Referenz-Schicht:
         2.1 Für jede Geometrie G in der thematischen Karte:
               WENN Art der topo. Beziehung zwischen R und G DANN
                    Merke TRUE für die topo. Beziehung für R
         2.2 WENN topologische Beziehung = TRUE DANN
               Füge TRUE für Geometrie R ein, sonst FALSE
         2.3 Gehe zu Punkt 2.
      3. Rückgabe der Referenz-Schicht um die topologische Beziehungen
```

Bemerkungen zum Algorithmus:

In Oracle werden Methoden bereitgestellt, die die topologischen Beziehungen zwischen geometrischen Objekten berechnen können. Dazu ist in *Oracle Spatial* das *9-Intersection-Modell* implementiert. Die Untersuchung der topologischen Beziehung zwischen zwei geometrischen Objekten erfolgt unter Benutzung der in Oracle definierten 12 Beziehungen, den sogenannten *Masken* [Tie03] [ORAb]. Diese Masken lassen sich mit dem Operator `SDO_RELATE` oder der Funktion `RELATE` aus dem `SDO_GEOM`-Package kombinieren, um die topologischen Beziehungen zwischen zwei geometrischen Objekten zu ermitteln. Eine weiterführende Analyse der Problematik der topologischen Beziehungen wird in [Tie03] durchgeführt.

5.3.6 Materialisierung von metrischen Beziehungen

Die Integration von metrischen Beziehungen ermöglicht die Einbeziehung der Entfernung. Die topologischen Beziehungen repräsentieren durch die Materialisierung lediglich ein Vorkommen in

der Umgebung, indem die Beziehung zwischen zwei geometrischen Objekten entweder TRUE oder FALSE ist. Die metrische Beziehung ist in diesem Punkt aussagekräftiger, z. B. die Universität ist 1 km vom Stadtzentrum entfernt. Im Vergleich drückt die topologische Beziehung bestenfalls *Universität berührt Stadtzentrum* aus.

Im Folgende wird ein Algorithmus angegeben, der die Transformation von metrischen Beziehungen bewirkt. Die Vorgehensweise ist analog zur Transformation der topologischen Beziehung (siehe Abb. 5.25), d. h. es wird wiederum eine Referenz-Schicht ausgewählt und die Schichten (thematische Karten) werden entsprechend der Referenz-Schicht in Form eines Attributes materialisiert.

Beispiel: Es seien wiederum die Referenz-Schicht Städte und die zwei thematischen Karten Straßen und Flüsse (Abb. 5.25) gegeben. Die Materialisierung der metrischen Beziehungen erfolgt dadurch, dass eine Geometrie in der Referenz-Schicht ausgewählt wird und anschließend dasjenige Objekt mit dem Minimal-Abstand in der thematischen Karte identifiziert wird. Dieser Abstand wird als metrische Beziehung zwischen dem Referenz-Objekt und dem thematischen Objekt gespeichert.

```
EINGABE: 1. Referenz-Schicht (Tabelle)
         2. Schicht (Tabelle) - thematische Karte
AUSGABE: Referenz-Schicht erweitert um ein Attribut
METHODE:
      1. Erstelle eine neue Spalte in der Tabelle
         (Bezeichnung: metrische Beziehung und themat. Karte)
      2. Für jede Geometrie R in der Referenz-Schicht:
         2.1 Für jede Geometrie G in der thematischen Karte:
               Berechne Abstand zwischen R und G
               WENN Abstand kleiner als Minimal-Abstand DANN
                   Merke neuen Abstand zwischen R und G als Minimal-Abstand
         2.2 Füge Minimal-Abstand für Geometrie R ein
         2.3 Gehe zu Punkt 2.
      3. Rückgabe der Referenz-Schicht (Tabelle) um metrische Beziehung
```

Bemerkungen zum Algorithmus:

In Oracle werden Methoden bereitgestellt, die bei der Materialisierung der metrischen Beziehung verwendet werden können. Es handelt sich um geometrische Funktionen, die im Folgenden aufgeführt sind:

- SDO_NN – Bestimmt die n nächsten Nachbarn einer Geometrie
- SDO_NN_DISTANCE – Bestimmt die Abstände derjenigen Objekte, die vom Operator SDO_NN zurückgeliefert werden
- SDO_WITHIN_DISTANCE – Bestimmt, ob zwei Geometrien innerhalb einer vorgegeben Distanz liegen
- SDO_DISTANCE – Berechnet minimalen Abstand zwischen zwei Geometrien

Weitere Informationen der geometrischen Objekte, die ebenfalls metrischer Natur sind, z. B. Länge eines Flusses oder Straße bzw. die Fläche eines Waldgebiets, können durch Funktionen, die bereits in [ORAb] integriert sind, in Form von Attributen materialisiert werden.

- SDO_AREA – Bestimmt die Fläche eines zweidimensionalen Polygons
- SDO_LENGTH – Bestimmt die Länge oder den Durchmesser einer Geometrie

5.3.7 Materialisierung von gerichteten Beziehungen

Die Integration von gerichteten Beziehungen vervollständigt die impliziten räumlichen Beziehungen, indem die relative Lage zweier Objekte zueinander ausgedrückt wird. Obwohl die Einbeziehung in das Spatial Data Mining, sowie die Materialisierung von gerichteten Beziehungen in der Literatur selten Anwendung findet, können gerichtete Beziehungen durchaus wertvolle Informationen beitragen.

In Abb. 5.26 (links) ist die Berechnung einer gerichteten Beziehung zwischen zwei geometrischen Objekten (Objekt A zu Objekt B über den Winkel α und Objekt A zu Objekt C über den Winkel β) graphisch skizziert. Die Beziehung definiert sich letztliche nach der Methodik (siehe Abb. 5.26 (rechts)), in der die gerichteten Beziehungen unterteilt sind (z. B. 4-Modell oder 8-Modell).

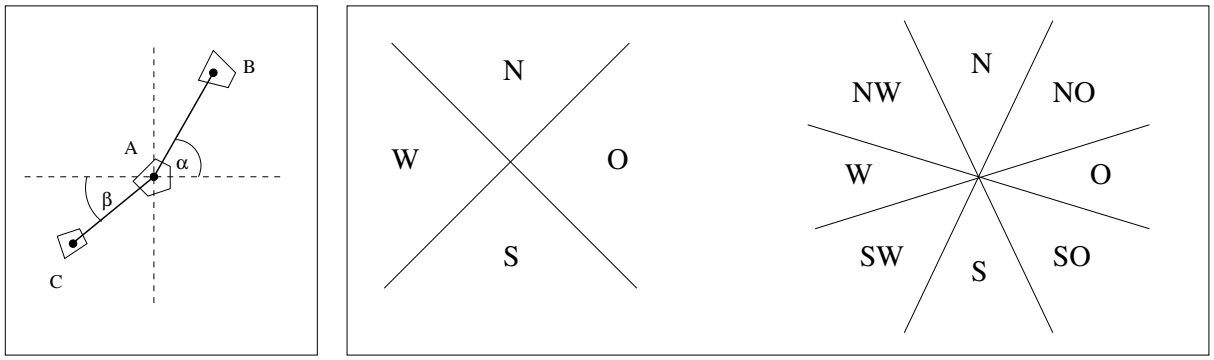


Abbildung 5.26: Materialisierung gerichteter Beziehungen

Im Folgenden wird ein Algorithmus angegeben, der eine Materialisierung von gerichteten Beziehungen realisiert. Das Vorgehen ist dabei analog zur Materialisierung von topologischen und metrischen Beziehungen (siehe Abb. 5.25).

```
EINGABE: 1. Referenz-Schicht (Tabelle)
         2. Schicht (Tabelle) - thematische Karte
         3. Methodik (Einteilung in 4 oder 8 Richtungen)
AUSGABE: Referenz-Schicht erweitert um ein Attribut
METHODE:
      1. Erstelle eine neue Spalte in der Tabelle
      2. Für jede Geometrie R in der Referenz-Schicht
         2.1 Berechne die Abstände von R mit jeder Geometrie G
             in thematischer Karte
         2.2 WENN Anzahl der Geometrie mit minimalen Abstand > 0 DANN
             2.2.1 Wähle beliebige Geometrie G zur Berechnung
         2.3 Berechne gerichtete Beziehung zwischen R und G
         2.4 Füge gerichtete Beziehung nach angegebener Methodik ein
      3. Rückgabe der Referenz-Schicht (Tabelle) um gerichtete Beziehung
```

Bemerkungen zum Algorithmus:

In Oracle werden Methoden bereitgestellt, die bei der Materialisierung von gerichteten Beziehungen eingesetzt werden können. Dabei handelt es sich um die gleichen geometrischen Funktionen, die bereits bei der Materialisierung von metrischen Beziehungen genutzt wurden.

6 Implementierung

In diesem Abschnitt wird die vorgenommene Implementierung des im vorigen Abschnitt 5.3.5 entwickelten Konzepts zur Materialisierung von räumlichen Daten und Beziehungen vorgestellt.

Die Implementierung wurde vollständig in PL/SQL vorgenommen und die im Folgenden beschriebenen Prozeduren und Funktionen sind im PL/SQL-Package `dbms_sdm_transform` zusammengefasst. Dabei werden ausschließlich die zur Benutzung relevanten Funktionalitäten vorgestellt. Eine Übersicht über alle implementierten Funktionen ist anschaulich in Abb. 6.1 dargestellt.

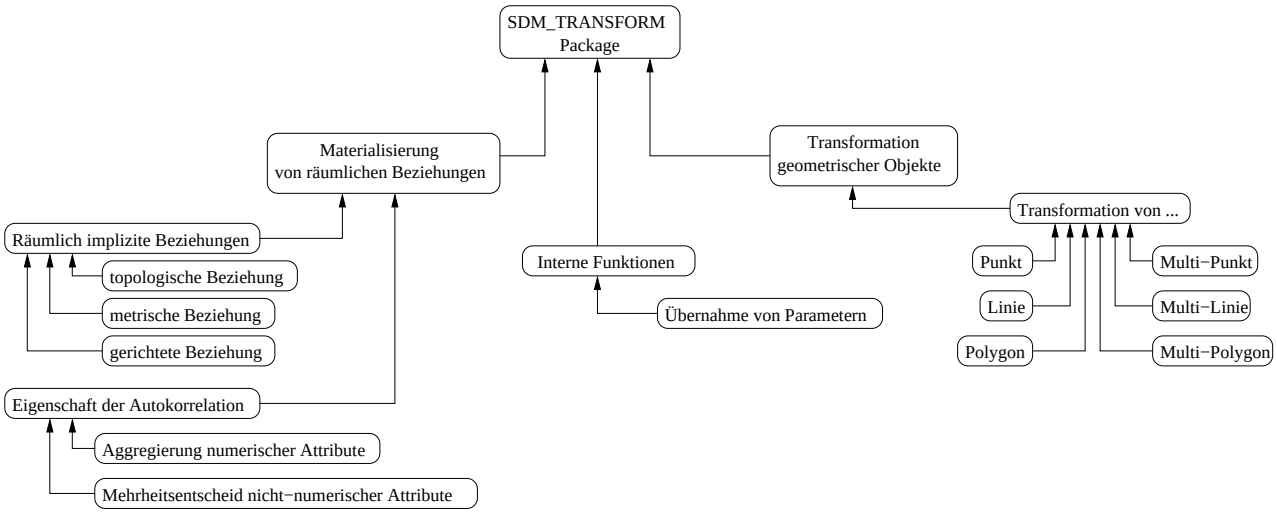


Abbildung 6.1: `dbms_sdm_transform`-Package

In Abb.6.2 ist die Integration des `dbms_sdm_transform`-Package für die Materialisierung von räumlichen Objekten und Beziehungen in das DBMS von Oracle dargestellt. Die Verbindungen zwischen den Elementen repräsentieren dabei direkte Beziehungen zwischen den Packages.

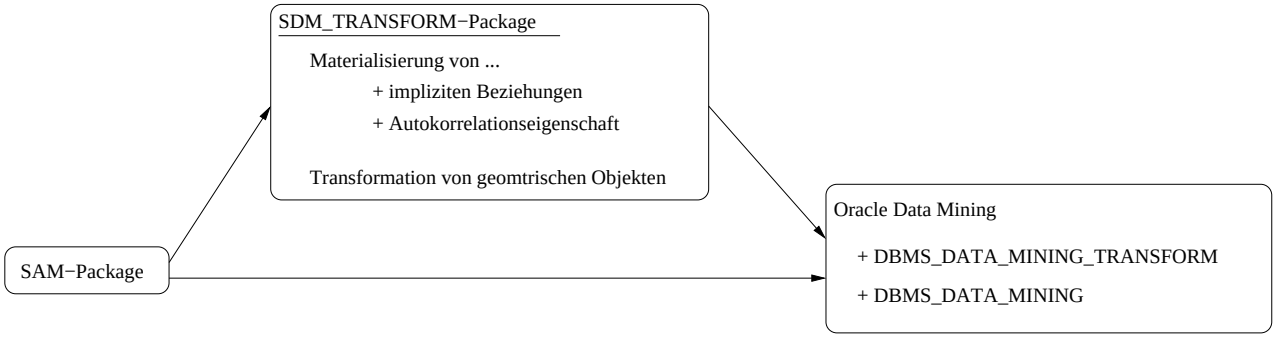


Abbildung 6.2: Integration des `dbms_sdm_transform`-Package in das Oracle DBMS

Weiter ist in Abb. 6.3 die Integration und der Ablauf der Materialisierung von räumlichen Objekten und Beziehungen in den KDD-Prozess für das Spatial Data Mining schematisch dargestellt.

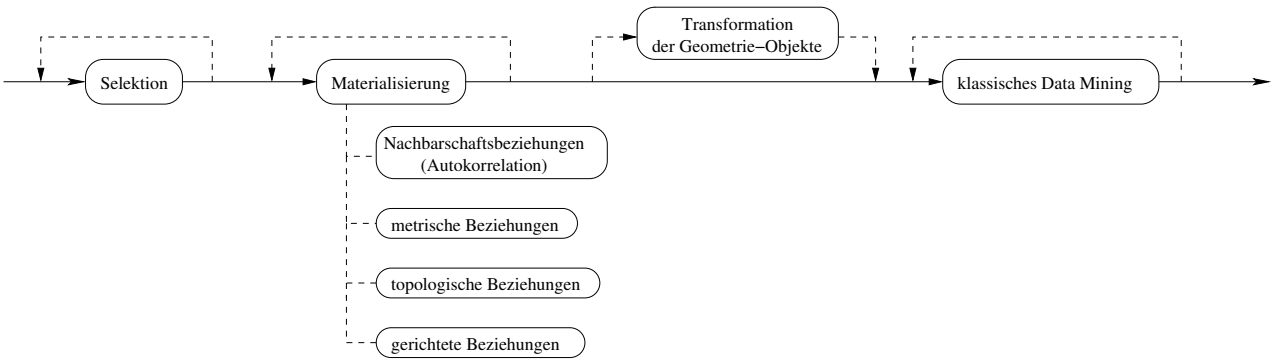


Abbildung 6.3: Integration der Materialisierung in den KDD-Prozess in räumlichen Datenbanken

In Abschnitt 6.1 wird die Implementierung der Prozeduren und Funktionen beschrieben. Anschließend wird in Abschnitt 6.2 das Spatial Data Mining unter Anwendung der implementierten Materialsierungsfunktionen und der Oracle Data Mining-Cartridge beispielhaft vorgestellt.

6.1 Spatial Data Mining-Transformation

Die Implementierung des PL/SQL-Packages `dbms_sdm_transform` richtet sich dabei nach dem Konzept und der Strukturierung des Preprocessing und der Transformation in der Oracle Data Mining-Cartridge.

Die Umsetzung der Materialisierung wird in zwei Kategorien (siehe Abb. 6.3) unterteilt:

1. Materialisierung von räumlichen Beziehungen
2. Materialisierung von geometrischen Objekten

6.1.1 Materialisierung von räumlichen Beziehungen

Um die Materialisierung von räumlichen Beziehungen nach dem Prinzip der Oracle Data Mining-Cartridge durchzuführen, sind die folgenden Schritte notwendig:

1. Erstellen einer Tabelle für die Materialisierung einer räumlichen Beziehung
2. Durchführung der Materialisierung
3. Anfügen der materialisierten Informationen an die Objekt-Tabelle

Für die aufgelisteten Schritte zur Materialisierung von räumlichen Beziehungen sind entsprechende Prozeduren und Funktionen im PL/SQL-Package `dbms_sdm_transform` implementiert.

Eine Materialisierung wird mit der folgenden Prozedur eingeleitet, wodurch eine Tabelle mit dem Namen *definition_table* konstruiert wird, die die Spalten *ID* und *Name_of_Materialisation* besitzt. Des Weiteren beschreibt *Type_of_Materialisation* die Art der Materialisierung und besitzt einen der folgenden Werte:

- `TOPO_REL` – topologische Beziehung
- `METRIC_REL` – metrische Beziehung
- `DIRECT_REL` – gerichtete Beziehung
- `NUM_AC_REL` – Eigenschaft der Autokorrelation von numerischen Attributen
- `CAT_AC_REL` – Eigenschaft der Autokorrelation von nicht-numerischen Attributen

In dieser Tabelle werden entsprechend die Identifikation der Objekte aus der Objekt-Tabelle und die materialisierte räumliche Beziehung zwischen diesen Objekten und weiteren geometrischen Objekten, der thematischen Karten, gespeichert.

```
PROCEDURE createRelMaterialisation( definition_table STRING,
                                    Name_of_Materialisation STRING,
                                    Type_of_Materialisation STRING )
```

Für die Materialisierung der topologischen, metrischen, gerichteten und Nachbarschaftsbeziehungen werden die nachfolgenden Prozeduren verwendet. Das Vorgehen dieser Prozeduren ist dabei miteinander vergleichbar. Daher ist in Abb. 6.4 die Materialisierung in diesem Schritt beispielsweise anhand der Materialisierung einer topologischen Beziehung mit der Bezeichnung `Topo_Materialisierung` dargestellt. Das Resultat unterscheidet entsprechend nach der Art der Materialisierung.

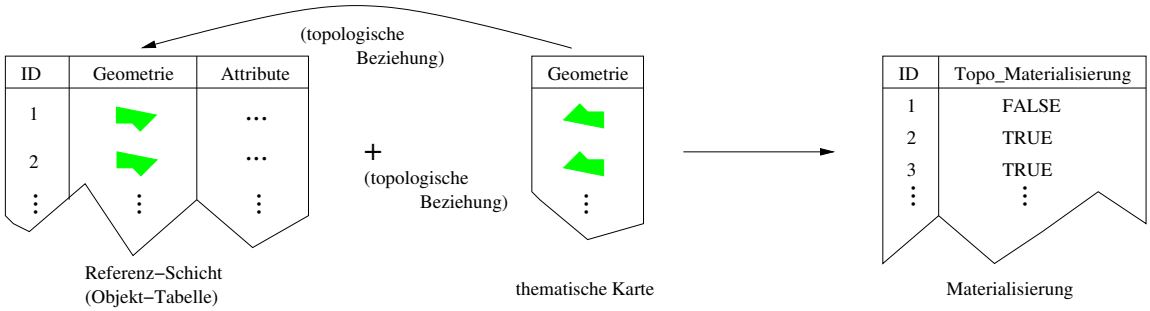


Abbildung 6.4: Materialisierung einer topologischen Beziehung

Die Prozedur `insertTopoRelMaterialisation` materialisiert eine topologische Beziehung durch Angabe einer Referenz-Schicht, die durch die Parameter Objekt-Tabelle, den Primärschlüssel und dem Geometrie-Objekt beschrieben wird, sowie einer thematischen Karte mit Geometrie-Objekt und einer geltenden topologischen Beziehung. Die topologische Beziehung wird dabei durch Angabe eines gültigen Bezeichners des *9-Intersection-Modells* von *Spatial Oracle* [ORAb] spezifiziert, z. B. `'mask=contains'` für `enthält` oder `'mask=inside+touch'` für `innerhalb` oder `berührt`.

```
PROCEDURE insertTopoRelMaterialisation (
          definition_table STRING, -- Tabelle für berechnete Werte
          object_layer STRING,     -- Objekt-Tabelle
          object_id STRING,        -- Spalte des Primärschlüssels,
          object_geom STRING,      -- Spalte der Geometrie
          theme_layer STRING,      -- thematische Karte
          theme_geom STRING,       -- Spalte der Geometrie
          topo_spec STRING)        -- topologische Beziehung
```

Die Prozedur `insertMetricRelMaterialisation` materialisiert eine metrische Beziehung durch Angabe einer Referenz-Schicht, die durch die Parameter Objekt-Tabelle, den Primärschlüssel und dem Geometrie-Objekt beschrieben wird, sowie einer thematischen Karte mit Geometrie-Objekt.

```
PROCEDURE insertMetricRelMaterialisation (
          definition_table STRING, -- Tabelle für berechnete Werte
          object_layer STRING,     -- Objekt-Tabelle
          object_id STRING,        -- Spalte des Primärschlüssels,
```

```
          object_geom STRING,      -- Spalte der Geometrie
          theme_layer STRING,      -- thematische Karte
          theme_geom STRING)       -- Spalte der Geometrie
```

Die Prozedur `insertDirctRelMaterialisation` materialisiert eine gerichtete Beziehung. Das Vorgehen und die Parameter entsprechen den der Prozeduren `insertTopoRelMaterialisation` und `insertMetricRelMaterialisation`. Ferner spezifiziert der Parameter *direct_modell* die Einteilung der Richtungen und besitzt nach Abb. 5.25 in Kapitel 5.26 einen der folgenden Werte: `4DM` oder `8DM` für 4 bzw. 8 Richtungen.

```
PROCEDURE insertDirctRelMaterialisation (
          definition_table STRING, -- Tabelle für berechnete Werte
          object_layer STRING,     -- Objekt-Tabelle
          object_id STRING,        -- Spalte des Primärschlüssels,
          object_geom STRING,      -- Spalte der Geometrie
          theme_layer STRING,      -- thematische Karte
          theme_geom STRING,       -- Spalte der Geometrie
          direct_modell STRING)    -- Modell zur Einteilung
```

Die Prozedur `insertNumACRelMaterialisation` materialisiert die Autokorrelationseigenschaft eines numerischen Attributs (`aggr_column` und `aggr_string`) durch Einbeziehung der Nachbarschaft (`theme_layer` und `theme_geom`) um eine gegebene Geometrie (`object_layer` und `object_geom`), die durch einen Bereich (`dst_spec`) spezifiziert ist.

```
PROCEDURE insertNumACRelMaterialisation (
          definition_table STRING, -- Tabelle für berechnete Werte
          object_layer STRING,     -- Objekt-Tabelle
          object_id STRING,        -- Spalte des Primärschlüssels
          object_geom STRING,      -- Spalte der Geometrie
          theme_layer STRING,      -- thematische Karte
          theme_geom STRING,       -- Spalte der Geometrie
          aggr_string STRING,      -- Aggregierungsfunktion
          aggr_column STRING,      -- Spalte für Aggregierung
          dst_spec STRING)         -- Angabe der Umgebung
```

Die Prozedur `insertCatACRelMaterialisation` materialisiert die Autokorrelationseigenschaft eines nicht-numerischen Attributs. Überdies sind die Beschreibung und die Parameter mit der Prozedur `insertNumACRelMaterialisation` übereinstimmend.

```
PROCEDURE insertCatACRelMaterialisation (
          definition_table STRING, -- Tabelle für berechnete Werte
          object_layer STRING,     -- Objekt-Tabelle
          object_id STRING,        -- Spalte des Primärschlüssels
          object_geom STRING,      -- Spalte der Geometrie
          theme_layer STRING,      -- thematische Karte
          theme_geom STRING,       -- Spalte der Geometrie
          aggr_column STRING,      -- Spalte für Aggregierung
          dst_spec STRING)         -- Angabe der Umgebung
```

Der Abschluss einer Materialisierung von räumlichen Beziehungen wird durch die folgende Prozedur vorgenommen, wobei eine neue Tabelle *target_table* durch das Anfügen der materialisierten Informationen (*definition_table*) an die Objekt-Tabelle *object_layer* konstruiert wird.

```
PROCEDURE xformRelMaterialisation (
            definition_table STRING, -- Materialisierung
            object_layer STRING,     -- Objekt-Tabelle
            object_id STRING,        -- Spalte des Primärschlüssels
            target_table  STRING)    -- Erweiterte Tabelle
```

6.1.2 Materialisierung von geometrischen Objekten

Für die Materialisierung von geometrischen Objekten[1] nach dem in Abschnitt 5.3.3 vorgestellten Konzept sind ebenfalls entsprechende Prozeduren und Funktionen im PL/SQL-Package `sdm_transform` implementiert.

Mit der folgenden Prozedur wird die Materialisierung von geometrischen Objekten eingeleitet. Dabei wird eine Tabelle mit dem Namen *augment_table_name* erstellt, deren Struktur sich nach der Struktur der Objekt-Tabelle (*object_table*) richtet, jedoch um die Spalten vom Typ `SDO_GEOMETRY` reduziert und um die Spalten X und Y erweitert.

```
PROCEDURE create_SDM_Transform (
            augment_table_name STRING, -- erweiterte Tabelle
            object_table STRING,       -- Objekt-Tabelle
            setting_table  STRING)     -- Konfigurationstabelle
```

Über die Konfigurationstabelle (*setting_table*), die beim Aufruf der Prozedur `create_SDM_Transform` angegeben werden kann, können die folgenden Parameter, die bei der Materialisierung der geometrische Objekte in eine repräsentative Punktmenge relevant sind, beeinflusst werden:

- `DBMS_SDM_TRANSFORM.MAX_AREA` – max. Größe eines Polygons für die Repräsentation durch einen einzelnen Centroiden
- `DBMS_SDM_TRANSFORM.MIN_AREA` – min. Größe eines Polygons für die Zerlegung nach einem Raster
- `DBMS_SDM_TRANSFORM.TOLERANCE` – max. Differenz zwischen der Fläche des Polygons und der Fläche der konvexen Hülle des Polygons
- `DBMS_SDM_TRANSFORM.MAX_LENGTH` – max. Länge einer Linie-Geometrie für die Repräsentation durch Start- und Endpunkt
- `DBMS_SDM_TRANSFORM.DIM_INFO` – Spezifiziert den Toleranzwert für Prozeduren im PL/SQL-Package `SDO_GEOM` [Ora06f]. Dieser Wert beschreibt den max. Abstand zwischen zwei Objekten, so dass diese noch als ein Objekt interpretiert werden.
- `DBMS_SDM_TRANSFORM.SCALE` und `DBMS_SDM_TRANSFORM.PRECISION` – Werte für die Angabe der transformierten (x, y)-Koordinaten

Das Schema der Konfigurationstabelle ist folgendermaßen:

Spaltenname	Datentyp	Bedeutung
setting_name	VARCHAR2(50)	Name des Parameters
setting_value	NUMBER(20,15)	Wert des Parameters

[1] Bei der Materialisierung von geometrischen Objekten handelt es sich vielmehr um eine Transformation, da lediglich beliebige geometrische Objekte vom Typ `SDO_GEOMETRY` in einen Punkt bzw. Punktmenge umgewandelt werden.

Mit der nachfolgenden Prozedur wird die Materialisierung der Geometrien in eine repräsentative Punktmenge durchgeführt. Die Prozedur speichert das Ergebnis der Transformation sofort in der erweiterten Tabelle (augment_table_name) ab. Dabei liefert die Objekt-Tabelle zum einen die Geometrie-Objekte (*object_geom*) und zum anderen die nicht-räumlichen Attribute, die in die erweiterte Tabelle, zu den transformierten Punkten, übernommen werden sollen – *unverändert* in der Liste *include_columns* und *aufgeteilt* in der Liste *partial_columns* vermerkt.

```
PROCEDURE insert_SDM_Transform (
            augment_table_name STRING, -- erweiterte Tabelle
            object_table STRING,       -- Objekt-Tabelle
            object_id STRING,          -- Spalte des Primärschlüssels
            object_geom STRING,        -- Spalte der Geometrie
            include_columns DBMS_DATA_MINING_TRANSFORM.Column_List,
                                       -- Spalten, deren Werte unverändert
                                       -- übernommen werden sollen
            partial_columns DBMS_DATA_MINING_TRANSFORM.Column_List
                                       -- Spalten, deren Werte prozentual
                                       -- aufgeteilt werden sollen
)
```

Der Abschluss der Materialisierung der geometrischen Objekte wird mit der folgenden Prozedur durchgeführt. In diesem Schritt wird die erweiterte Tabelle um einen eindeutigen Identifikator ergänzt, der durch die Transformation der Geometrien (z. B. Linie wird durch Start- und Endpunkt repräsentiert) in die Punktdarstellung verloren gegangen sein könnte.

```
PROCEDURE xform_SDM_Transform (augment_table_name STRING)
```

Das Resultat der Transformation der Geometrie-Objekte in eine repräsentative Punkt-Darstellung ist in Abb. 6.5 dargestellt. Dabei ist das Attribut `Geometrie` in den Attributen `X` und `Y` aufgegangen, die Zugehörigkeit einer Punkt-Darstellung zu einem Geometrie-Objekt im ursprünglichen Attribut `ID` (Identifikation der Daten) gespeichert und eine neue Identifikation durch das Attribut `SID` angefügt.

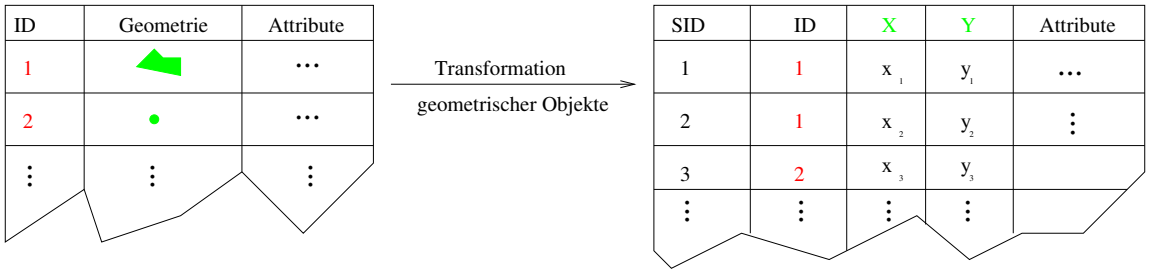


Abbildung 6.5: Transformation der Geometrie-Objekte in eine Punkt-Darstellung

6.2 Nutzen von Materialisierungen für das Spatial Data Mining

Nachdem mögliche Transformationen für implizite räumliche Beziehungen und Transformationen für geometrische Objekte vorgestellt worden sind, stellt sich im Folgenden die Frage, bei welchen Data Mining-Methoden welche Transformationen eingesetzt werden können. Um diese Frage zu

klären, sollen Beispiel-Anwendungen aus dem Bereich GIS (Geographische Informationssysteme) analysiert werden.

Bemerkung:

In den folgenden Beispielen wird davon ausgegangen, dass das PL/SQL-Package `dbms_sdm_transform` für die Materialisierungen von geometrischen Objekten und räumlichen Beziehungen zur Verfügung steht. Weiterhin wird angenommen, dass entsprechende Metadaten und räumliche Index für die Tabellen vorhanden sind.

6.2.1 Fallbeispiel: Clustering von Ballungsgebieten

Das erste Fallbeispiel beschäftigt sich mit der Problemstellung der Identifizierung von Ballungsgebieten. Die erforderlichen Bevölkerungsdaten liefert das Gazetteer-Projekt [Gaz06], das zahlreiche Daten für Staaten und Siedlungen (Siedlungsgebiete und Städte) bereitstellt (siehe Abb. 6.6).

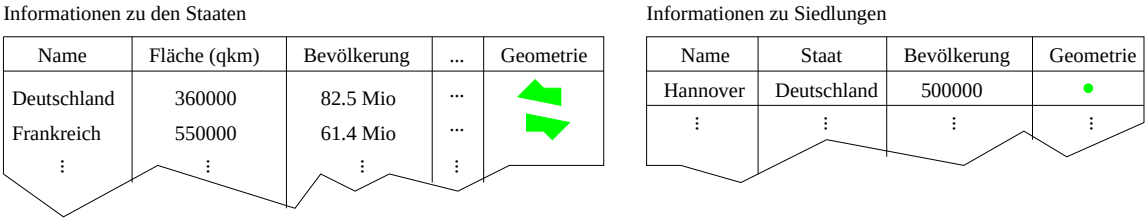


Abbildung 6.6: Daten aus dem Gazetteer-Projekt [Auszug]

Lösungsansatz:

Der intuitive Lösungsansatz zur Beantwortung des Problems für das Clustering von Ballungsgebieten besteht darin, dem Algorithmus die Attribute `Längengrad`, `Breitengrad` und `Bevölkerung` bereitzustellen und das Clustering anhand dieser Attribute durchzuführen. Dieses Vorgehen hat jedoch folgende Schwachstellen:

- Bei der Berechnung der Ähnlichkeit mit dem euklidischen Abstand kann es durch die unterschiedlichen Skalen der Attribute zu einer unvorhergesehenen Dominanz kommen, z. B. bei den Wertebereichen Längengrad ($-180° \leq l \leq 180°$), Breitengrad ($-90° \leq b \leq 90°$) und Bevölkerung ($population \gg 0$) wird die Ähnlichkeit durch die Bevölkerung dominiert.
- Darüber hinaus können Siedlungen oder Städte zu Cluster zusammengefasst werden, die zwar ähnlich zueinander sind und eine für das Clustering notwendig Häufung bilden, jedoch keinem Ballungsgebiet entsprechen. Das Ergebnis liefert demnach viele Cluster, worunter auch keine Ballungsgebiete sind.

Mögliche Maßnahmen zur Behebung der angesprochenen Probleme und letztlich zur Lösung des Clustering von Ballungsgebieten sind im Folgenden aufgelistet:

- Eine Möglichkeit besteht in der Normierung der Attribute nach Kapitel 2.5 (eine eventuelle existierende Dominanz wird dadurch beseitigt) und anschließender Gewichtung der Bevölkerung im Vergleich zu den Attributen Längengrad und Breitengrad. Dabei stellt sich jedoch die entscheidende Frage: Wie findet man eine geeignete Gewichtung der Attribute?
- Eine weitere Möglichkeit besteht in der Definition des Ballungsgebiets und anschließendem Ausschluss aller Bereiche, die die definierten Kriterien nicht erfüllen.

Lösung des Problems:

Im Folgenden soll der zweite Lösungsansatz verfolgt werden, da selbst durch die Definition einer geeigneten Gewichtung das Problem weiterbesteht, dass Cluster von Regionen gefunden werden, die keine Ballungsgebiete sind. (Der Quellcode zum Beispiel ist in Anhang A.1 nachzulesen.)

Im ersten Schritt erfolgt die Vorverabeitung der relevanten Daten für das Clustering von Ballungsgebieten. Dazu muss zunächst der Begriff des *Ballungsgebiets* definiert werden:

> Ein *Ballungsgebiet* ist eine Kernstadt, die ein suburbanes Umland oder zumindest dicht besiedeltes Gebiet besitzt, das außerhalb der Stadtgrenzen liegt, aber direkt an sie angrenzt. Ein Ballungsgebiet kann sich dabei aus einer oder mehreren Städten und ihren Vorstadtgemeinden zusammensetzen.[WIK06] Weiterhin muss die Bevölkerungszahl von 1000000 Einwohnern und eine Bevölkerungsdichte von mind. l000 Einwohnern pro km^2 gegeben sein.

Für die Selektion der relevanten Daten sind demnach die Bevölkerungsdichte und die Bevölkerungszahl von einer Region und seinen Nachbarn von Interesse. Aufgrund der gegebenen Daten, stehen jedoch lediglich die Bevölkerungszahlen von Siedlungen (Siedlungsgebiete und Städte) zur Verfügung.

Um die Bevölkerungsdichte und Bevölkerungszahl zu ermitteln, sind die folgenden Schritte durchzuführen. Zunächst wird ein feines Raster (`demo_raster`) definiert und über die Karte mit den Siedlungen gelegt. Dadurch werden Regionen beschrieben, deren Bevölkerung sich aus der Summation der darin befindlichen Siedlungen ergibt. In Abb. 6.7 (a) ist beispielhaft ein Raster dargestellt.

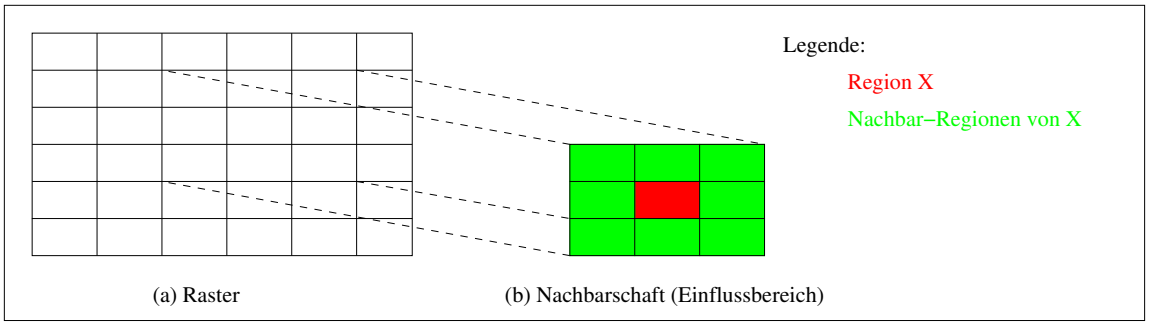


Abbildung 6.7: Definition: Raster und Nachbarschaft

In Listing 6.1 ist die Definition eines Rasters nach Abb. 6.7 (a) mittels einer Funktion aus dem `SAM`-Package abgebildet [Ora06g]. Die Parameter für diese Funktion sind die Größe des Bereichs (x-Ausdehnung $\in [5.865, 15.033]$ und y-Ausdehnung $\in [47.274, 55.056]$), sowie der Grad der Zerlegung des Bereichs (Grad 6 bedeutet, dass jede Dimension in $2^6 = 64$ Abschnitte unterteilt wird).

```
CREATE TABLE demo_raster AS
 SELECT *
 FROM TABLE(SDO_SAM.TILED_BINS(5.865, 15.033, 47.274, 55.056, 6));
```

Listing 6.1: SQL-Anfrage zur Definition eines Rasters

Die Summation der Bevölkerung für die jeweilige Region wird dabei durch die Materialisierung von Nachbarschaftsbeziehungen (Autokorrelation), d. h. die Raster-Geometrie im Bezug auf die dazugehörigen Siedlungen, realisiert. In Listing 6.2 wird dieser Vorgang unter Verwendung der Funktionen aus dem `dbms_sdm_transform`-Package durchgeführt und das Ergebnis steht im Anschluss in der Tabelle `demo_agglo_1` bereit. Die Materialisierung ist im Attribut `aggr_population` gespeichert.

```
BEGIN
   DBMS_SDM_TRANSFORM.createRelMaterialisation(
          'def_materialisation_1','aggr_population','NUM_AC_REL');
```

```
   DBMS_SDM_TRANSFORM.insertNumACRelMaterialisation(
          'def_materialisation_1','demo_raster','id','geometry',
          'data_cities','sdo_location','SUM','population',NULL);
   DBMS_SDM_TRANSFORM.xformRelMaterialisation(
          'def_materialisation_1','demo_raster','id','demo_agglo_1');
END;
```

Listing 6.2: Summation der Bevölkerung die definierten Regionen

Durch diese Transformation kann die zweite Bedingung, die Dichte, der Definition eines Ballungsgebiets berechnet werden, indem der Wert der aggregierten Bevölkerung durch die Fläche des Raster-Elements geteilt wird.

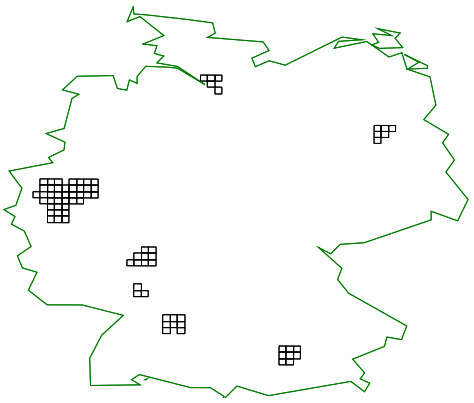
Die erste Bedingung, d. h. mind. 1000000 Einwohner, kann nicht einfach mit dem Wert der aggregierten Bevölkerung berechnet werden. Der Ursache liegt darin, dass wenn die Raster-Elemente zu fein sind, der Wert der aggregierten Bevölkerung ausschließlich bei Metropolen den Grenzwert überschreitet. Aus diesem Grund werden zu jedem Raster-Element die Nachbar-Regionen nach Abb. 6.7 (b) betrachtet und dadurch die Eigenschaft der Autokorrelation, d. h. benachbarte Objekte beeinflussen sich gegenseitig, berücksichtigt.

Diese Materialisierung der Nachbarschaftsbeziehungen ist auf die nächsten 8 Nachbarn (siehe Abb. 6.7 (b)) beschränkt und mit den Funktionen aus dem `dbms_sdm_transform`–Package in Listing 6.3 dargestellt.

```
BEGIN
 DBMS_SDM_TRANSFORM.createRelMaterialisation(
  'def_materialisation_2','aggr_pop_neighbors','NUM_AC_REL');
 DBMS_SDM_TRANSFORM.insertNumACRelMaterialisation(
  'def_materialisation_2','demo_raster','id','geometry',
  'demo_agglo_1','geometry','SUM','aggr_population','sdo_num_res=8');
 DBMS_SDM_TRANSFORM.xformRelMaterialisation(
  'def_materialisation_2','demo_agglo_1','id','demo_agglo_2');
END;
```

Listing 6.3: Materialisierung der Informationen über die Bevölkerung von 8 Nachbarn

Nach der Transformation können die Regionen selektiert werden, die der Definition eines Ballungsgebiets genügen, d. h. mind. 1000000 Einwohner und mind. 1000 Einwohner pro $1\,km^2$. (Die SQL-Anfrage zur Selektion der Daten wird hier nicht aufgeführt.)



In der nebenstehenden Abbildung ist Karte von Deutschland dargestellt, wobei ausschließlich die Regionen abgebildet sind, die die Kriterien erfüllen. Durch die zweidimensionale Darstellung ist bereits jetzt schon erkennbar, dass sich Regionen zu Cluster zusammenschließen. Aufschluss über die Cluster wird letztlich erst ein Clustering mit Oracle Data Mining geben.

Zum Abschluss der Transformation werden die zu Beginn definierten Regionen in x, y-Koordinaten materialisiert. Diese Materialisierung wird anhand der folgenden Befehle (siehe Listing 6.4) durchgeführt. Nach der Abarbeitung der Befehle kann die erweiterte Tabelle `ballungsgebiete` mit den Mechanismen in der Oracle Data Mining-Cartridge verarbeitet werden.

```
BEGIN
  DBMS_SDM_TRANSFORM.create_SDM_Transform('ballungsgebiete',
                                          'demo_agglo_2');
  DBMS_SDM_TRANSFORM.insert_SDM_Transform('ballungsgebiete',
                                          'demo_agglo_2',
                                          'geometry' );
  DBMS_SDM_TRANSFORM.xform_SDM_Transform('ballungsgebiete');
END;
```

Listing 6.4: Materialisierung mit dem SDM_TRANSFORM-Package

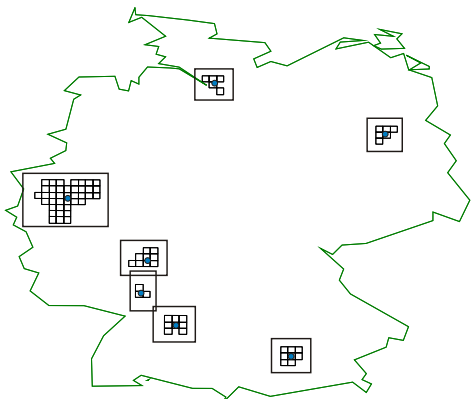
Im nächsten Schritt wird das Clustering mit dem klassischen Data Mining durchgeführt (siehe Kapitel 4.3.1), wobei im Folgenden lediglich nach den (x,y)-Koordinaten geclustert werden soll. In Listing 6.5 sind die entsprechenden Schritte für das Clustering mit dem Enhanced k-Means-Algorithmus zusammengefasst. (Ein Preprocessing und eine Transformation mit den Methoden in ODM wird nicht durchgeführt.)

```
-- Einstellung für das Clustering der Ballungsgebiete
--      1. 7 Cluster mit dem k-mean-Algorithmus
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
        model_name               => 'clus_model',
        mining_function          => DBMS_DATA_MINING.CLUSTERING,
        data_table_name          => 'ballungsgebiete'
        case_id_column_name      => 'id',
        settings_table_name      => 'clus_settings');
END;
```

Listing 6.5: Spatial Clustering mit Oracle Data Mining

Anmerkung: Aufgrund der Tatsache, dass mit dem Enhanced k-Means-Algorithmus maximal 20 Cluster bestimmt werden können und der O-Cluster-Algorithmus nicht funktioniert (siehe Kapitel 4), wird der Bereich des Clustering auf Deutschland beschränkt.

Das Clustering mit dem Enhanced k-Means-Algorithmus liefert, die in der nebenstehenden Abbildung, dargestellten Cluster. Die Centroide der Cluster sind blau eingefärbt und die Rechtecke repräsentieren die Cluster-Regeln. Dabei ist zu erkennen, dass die wichtigsten Ballungsgebiete in Deutschland durch den Enhanced k-Means-Algorithmus zu Cluster zusammengefasst wurde:



- Rhein-Ruhr
- Rhein-Main-Gebiet
- Rhein-Neckar-Dreieck
- Berlin, München, Hamburg
- Stuttgart

Anmerkung: Ein Clustering ohne jegliche vorherige Selektion würde keinen Sinn machen, da durch die Definition des Rasters und der anschließenden Aggregierung der Bevölkerung die (x,y)-Koordinaten der Centroiden gleichverteilt sind. Dadurch wäre nur noch ein Clustering nach der Dichte bzw. Bevölkerung möglich.

6.2.2 Fallbeispiel: Beziehungen zwischen Geo-Daten

Das zweite Beispiel orientiert sich an den Beispielen aus der Literatur zum Spatial Data Mining [KH95] [RT04] [VB06a]. Im Folgenden soll nach Räumlichen Assoziationsregeln gesucht werden, welche Wohngebiete, Straßen, Grünanlagen und Sportanlagen in Betracht zieht.

Im ersten Schritt werden zunächst die benötigten Schichten definiert, die für die Assoziationsanalyse von Interesse sind. In Listing 6.6 ist die Definition einer Schicht, die Straßen enthält, dargestellt.

```
CREATE TABLE nordseew_Strassen as
   select rownum as id, B.layerno as layerno, b.name as Bez,
          C.name as name, D.objectgeometry as shape
   from nordseew.objects A
        join nordseew.layer B on (A.layerno=b.layerno)
        join nordseew.object_race C on (A.objectraceno=C.objraceno)
        join nordseew.object_geometries D on (A.objno=D.objectno)
   where B.layerno = 104 and name = 'Strasse';
```

Listing 6.6: Definition einer Schicht für die Assoziationsanalyse

Im nächsten Schritt werden Materialisierungen mit dem `dbms_sdm_transform`-Package durchführt. Dabei bezeichnet die Schicht mit den Wohngebieten die Referenz-Schicht und die übrigen Schichten sind die thematischen Karten, die in Beziehung gesetzt werden. In Listing 6.7 - 6.9 sind die Materialisierungen der Informationen dargestellt, wobei die folgenden Attribute bzw. Beziehungen materialisiert werden sollen[2]:

- `ANYINTERACT_STREET` – beschreibt eine beliebige topologische Beziehung zwischen Wohngebiet und Straße
- `DIST_PARK` – beschreibt den kürzesten Abstand eines Wohngebiets zu einer Grünfläche
- `DIST_SPORT` – beschreibt den kürzesten Abstand eines Wohngebiets zu einer Sportanlage

```
-- Mater. einer topologischen Beziehung zw. Wohngebiet und Straße
-- tmp_topo_1 - Hilfstabelle zur Speicherung der Materialisierung
-- res_mat_1 - erweiterte Tabelle mit der Materialisierung
BEGIN
 DBMS_SDM_TRANSFORM.createRelMaterialisation(
                  'tmp_topo','ANYINTERACT_STREET','TOPO_REL');
 DBMS_SDM_TRANSFORM.insertTopoRelMaterialisation(
                  'tmp_topo','Wohngebiet','id','shape',
                  'Strassen','shape','mask=ANYINTERACT');
 DBMS_SDM_TRANSFORM.xformRelMaterialisation(
                  'Wohngebiet','id','tmp_topo','res_mat_1');
END;
```

Listing 6.7: Materialisierung der topologischen Beziehung zw. Wohngebiet und Straße

```
-- Mater. einer metrischen Beziehung zw. Wohngebiet und Grünfläche
BEGIN
 DBMS_SDM_TRANSFORM.createRelMaterialisation(
                  'tmp_metric_1','DIST_PARK','METRIC_REL');
```

[2]Die Benutzung der restlichen Funktionen aus dem `dbms_sdm_transform`-Package funktioniert analog.

```
 DBMS_SDM_TRANSFORM.insertMetricRelMaterialisation (
                 'tmp_metric_1','tmp_res_1','id','shape',
                 'Grünflächen','shape');
 DBMS_SDM_TRANSFORM.xformRelMaterialisation(
                 'res_mat_1','id','tmp_metric_1','res_mat_2');
END;
```

Listing 6.8: Materialisierung der metrischen Beziehung zw. Wohngebiet und Grünfläche

```
-- Mater. einer metrischen Beziehung zw. Wohngebiet und Sportanlage
BEGIN
 DBMS_SDM_TRANSFORM.createRelMaterialisation(
                 'tmp_metric_2','DIST_SPORT','METRIC_REL');
 DBMS_SDM_TRANSFORM.insertMetricRelMaterialisation (
                 'tmp_metric_2','res_mat_2','id','shape',
                 'Sportanlagen','shape');
 DBMS_SDM_TRANSFORM.xformRelMaterialisation(
                 'res_mat_2','id','tmp_metric_2','res_mat_3');
END;
```

Listing 6.9: Materialisierung der metrischen Beziehung zw. Wohngebiet und Grünfläche

Nach diesen Schritten besitzt die anfängliche Tabelle `Wohngebiet`, die lediglich Informationen zu Wohngebieten hatte, drei weitere Attribute, die Beziehungen über Straßen, Parkanlagen und Sportanlagen repräsentieren. In Abb. 6.8 ist diese Materialisierung anschaulich dargestellt.

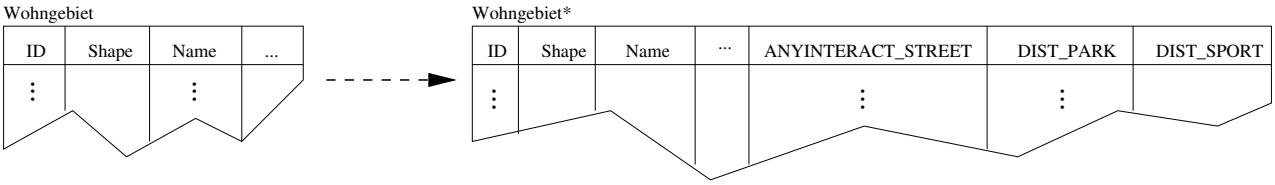


Abbildung 6.8: Materialisierung der metrischen Beziehung zw. Wohngebiet und Sportanlage

Im nächsten Schritt, nachdem alle relevanten Information zusammengestellt worden sind, kann die Assoziationsanalyse mit der Oracle Data Mining-Cartridge durchgeführt werden. Die Vorgehensweise richtet sich dabei nach dem in Kapitel 3 vorgestellten und in Kapitel 4.1.1 demonstrierten Prinzip.

Anmerkung:

> Eine Transformation der geometrischen Objekte in (x,y)-Koordinaten ist bei diesem Beispiel nicht durchzuführen, da die räumliche Position der Objekte bei der Assoziationsanalyse nicht von Bedeutung ist.

Im Transformationsschritt für die Assoziationsanalyse wird eine Diskretisierung der numerischen Attribute – speziell `DIST_PARK` und `DIST_SPORT` – durchgeführt. Im Anschluss wird dann die Assoziationsanalyse, wie in Listing 6.8 skizzenhaft dargestellt, realisiert.

```
BEGIN
  -- Diskretisierung von DIST_PARK und DIST_SPORT
  --jeweils in 4 Bereiche
```

```
  DBMS_DATA_MINING_TRANSFORM.CREATE_BIN_NUM('conf_asso_bin');
  DBMS_DATA_MINING_TRANSFORM.INSERT_BIN_NUM_QTILE(
              'conf_asso_bin','res_mat_3', 4,
    DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST('id'));
  DBMS_DATA_MINING_TRANSFORM.XFORM_BIN_NUM(
              'conf_asso_bin', 'res_mat_3', 'sdm_asso' );
END;
/
-- Modell erstellen
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
       model_name               => 'demo_asso_model',
       mining_function          => DBMS_DATA_MINING.ASSOCIATION,
       data_table_name          => 'sdm_asso',
       case_id_column_name      => 'id',
       settings_table_name      => 'demo_asso_dm_settings');
END;
```

Listing 6.10: Räumliche Assoziationsanalyse mit Oracle Data Mining

Das Ergebnis der Assoziationsanalyse sind dann Räumliche Assoziationsregeln, die beispielsweise die folgende Assoziation beschreibt. Dabei stellt `DIST_PARK=2` einen diskretisierten Wert dar und muss entsprechend zurückgerechnet werden, z. B. entspricht `DIST_PARK=2` einer Distanz von 500 m.

`Name='Wohngebiet'` $\wedge$ `DIST_PARK=2` $\Rightarrow$ `Strasse='FALSE'` [s=20%, c=80%]

Die Regel besagt, dass 80% der Wohngebiete, die in einer Entfernung von 2 (diskretisierte Wert) eine Grünanlage haben, keine Beziehung zu einer Straße besitzen.

Weitere Regeln dieser Art können mittels dieser Materialisierungen von räumlichen Beziehungen ermittelt werden, jedoch ergben dabei leider vielfach Trivialitäten, z. B. beschreibt die folgende Regel, dass Bahnhöfe von Schienen berührt werden (geometrisch betrachtet).

`Name='Bahnhof'` $\Rightarrow$ `berührt_Schiene='TRUE'` [s=20%, c=100%]

Die in diesem Abschnitt exemplarisch illustierten Beispiele zeigen bereits, dass durch die Materialisierung von räumlichen Beziehungen und geometrischen Objekten, mit dem PL/SQL-Package `dbms_sdm_transform`, Spatial Data Mining betrieben werden kann. Dabei gliedert sich das Package `dbms_sdm_transform` nahezu nahtlos in die Prozess-Struktur des KDD-Prozesses in Form eines weiteren Preprocessing ein.

7 Ausblick

Mit dieser Arbeit wurde ein Überblick über das Data Mining und den Knowledge Discovery-Prozess in *klassischen* relationalen Datenbanken gegeben. Diesbezüglich wurde eine konkrete Implementierung eines Data Mining-Systems, die Oracle Data Mining-Cartridge, vorgestellt und der darin bereitgestellte Funktionsumfang umfassend analysiert. Darüber hinaus wurde die Anwendung anhand von unterschiedlichen Beispielen demonstriert. Darauf aufbauend wurde ein Konzept entwickelt, das das Spatial Data Mining in räumlichen Datenbanken unter Verwendung dieser Methoden und Mechanismen ermöglichen soll. Daraus ergeben sich weitere Möglichkeiten in der Erweiterung sowie der Anwendung des Data Mining in räumlichen Datenbanken.

- **Erweiterung der Oracle Data Mining-Cartridge**
  Die Integration von Data Mining-Techniken in ein Datenbanksystem stellt einen wichtigen Schritt dar, die unterschiedlichen Disziplinen des Data Mining zu verbinden und somit eine komfortable Möglichkeit zur *alltäglichen* Benutzung von Data Mining-Verfahren für die Datenanalyse bereitzustellen. Aus diesem Grund steht bei der Oracle Data Mining-Cartridge steht primär der Aspekt der reinen Anwendung der vorhandenen Methoden für das klassische Data Mining im Vordergrund. Eine Erweiterung der implementierten Mechanismen zum Data Mining, z. B. die Integration einer Ähnlichkeitsmatrix für das Clustering o. ä., ist von Oracle nicht beabsichtigt. Daher wäre eine Schnittstelle für die Erweiterung dieser Cartridge um eigene neue Algorithmen sowohl für das klassische als auch das Spatial Data Mining wünschenswert.

- **Bewertung der Funktionen für die Materialisierung von räumliche Informationen**
  In dieser Arbeit wurden Funktionen zur Materialisierung von räumlichen Informationen, d. h. der topologischen, metrischen und gerichteten Beziehungen, sowie der Autokorrelation und der geometrischen Objekte, entwickelt. Aufgrund von Zeitmangel konnte jedoch nicht ausreichend geklärt werden, welche Problemstellungen des Spatial Data Mining dadurch realisiert werden können. In diesem Punkt besteht daher eine Notwendigkeit für weitere Untersuchungen in Form von Beispielen, um die Grenzen des Konzepts und der Realisierung zu bestimmen.

- **Erweiterung der Materialisierung**
  Das entwickelte und implementierte Konzept zur Materialisierung von räumlichen Beziehungen und Informationen orientiert sich an die speziellen Eigenschaften von räumlichen Daten und die in der Literatur beschriebenen Herausforderungen beim Spatial Data Mining. Dennoch wurde in dieser Arbeit lediglich eine Materialisierung mit einem einfachen Funktionsumfang realisiert. Folgende Erweiterungen sind daher denkbar:

  - Die Integration und die Verarbeitung von Konzept-Hierarchien bei der Materialisierung. Die bisherige Lösung muss den Umweg über die Definition von weiteren Schichten gehen, z. B. eine Schicht mit Gewässern wird unterteilt in 1. Schicht mit Flüssen 2. Schicht mit Seen und 3. Schicht mit künstlichen Gewässern.
  - Weiterhin werden bei der topologischen, metrischen und gerichteten Beziehung jeweils immer nur zwei Objekte bei der Materialisierung betrachtet. Eine Erweiterung dahinge-

hend das mehrere Objekte bei einer Materialisierung beteiligt sind, wäre ebenfalls wünschenswert.

- Ein anderer Aspekt betrifft die Aussagekraft der Materialisierung, beispielsweise wird bei der Materialisierung einer topologischen Beziehung lediglich ein `TRUE` oder `FALSE` markiert. Eine Erweiterung könnte beispielsweise die in Kapitel 5.2.3 vorgestellte Materialisierung nach dem *feature instance*-Prinzip sein.
- Die Selektion der relevanten Daten, die in dieser Arbeit jedoch nicht weiter behandelt wurde, stellt aufgrund der Autokorrelation von räumlichen Daten ein weiteres Problem für das Spatial Data Mining und entsprechend einer Materialisierung dar, das beachtet werden muss. Durch geeignete Methoden und Algorithmen sollte es daher möglich sein, die Signifikanz der Korrelation zu bewerten, um eine entsprechende Selektion für das Data Mining und dadurch eine Selektion von relevanten Materialisierungen bereitstellen zu können.

- **Spatial Data Mining ohne Materialisierung**
  Die Materialisierung von geometrischen Informationen ist ein guter Ansatz, um mit den Besonderheiten von räumlichen Daten und dem Spatial Data Mining umzugehen. Durch eine entsprechende Materialisierung besteht bereits die Möglichkeit implizite Beziehungen, sowie die Autokorrelation ausreichend abzubilden.

  Jedoch sind dieser Methode schnell Grenzen gesetzt, da entsprechende Materialisierungen einen enormen Mehraufwand in der Vorverarbeitung bedeuten. Aufgrund dessen ist das Vorgehen bei der Materialisierung in den vorliegenden Beispielen auf die Grundprinzipien beschränkt. Des Weiteren ist zu beachten, dass keine *perfekte* Materialisierung durchführbar ist, d. h. durch die Materialisierung kann immer nur ein gewisser Bruchteil abgebildet werden.

  Dadurch ergibt sich zwangsläufig die Notwendig der Entwicklung von Data Mining-Algorithmen, die speziell für das Spatial Data Mining konzipiert sind und mit den Besonderheiten von räumlichen Daten umgehen können. In Kapitel 5.2.2 wurde diesbezüglich die zweite Alternative beim Spatial Data Mining identifiziert, bei der sowohl das Data Mining als auch das Ergebnis im räumlichen Kontext betrachtet worden ist. Auf diese Weise könnte die Vorverarbeitung, wie sie bei der Materialisierung der Fall war, reduziert werden, wodurch das Hauptaugenmerk wieder auf das Data Mining gelegt werden kann. Die Lösungsansätze aus der Literatur (siehe Kapitel 5.2.2) schlagen bereits verschiedene Methodiken zum Spatial Data Mining vor.

  - Die Einführung von effizienten Datenstrukturen zur Abbildung von Nachbarschaften und die damit verbundene Integration in das Spatial Data Mining. Obwohl in [EKS97] bereits eine Datenstruktur und entsprechende Operationen definiert worden sind, stellt bei größeren Datenmengen der große Speicherbedarf ein erhebliches Problem dar.
    Eine weitere Integration eines Nachbarschaftsgraphen wurde in [Mor01] vorgestellt, um damit Co-Location Analyse durchzuführen. Der darin verwendete Graph, das Voronoi-Diagramm, ist einerseits effizient in der Konstruktion, dennoch kann dieser nicht für andere Objekte als für Punkte definiert werden.
  - Eine weitere interessante Alternative Spatial Data Mining ohne die Einführung von Datenstrukturen und ohne jegliche Materialisierung durchzuführen, besteht in der Abbildung der klassischen Problemstellungen auf die räumlichen Pendants und entsprechende Abbildungen für die Lösungsmethoden zu finden [RT04] [RT05]. Dabei ergeben sich die Schwierigkeiten bei der Abbildung, die ein im Vergleich wesentlich komplexeres Verständnis des Problems verlangt.

A Spatial Data Mining - Beispiele

Im Folgende ist der Quellcode des Beispiels zur Demonstration des `dbms_sdm_transform`-Packages in Kombination mit der Oracle Data Mining-Cartridge aufgeführt.

A.1 Clustering von Ballungsgebieten

```
CREATE TABLE data_cities AS
  SELECT * FROM gazetteer.gazetteer a
  WHERE a.country = 'Germany' AND a.admin_lvl3 is not null
    AND a.admin_lvl2 is not null AND a.admin_lvl1 is not null;

-- Grenzen für das Raster bestimmen
SELECT sdo_geom.sdo_mbr(geometry)
FROM gazetteer.country_9_1 where cntry_name like 'Germany%';

CREATE TABLE demo_raster AS
 SELECT *
 FROM TABLE(SDO_SAM.TILED_BINS(5.865003, 15.033816, 47.274714, 55.056528, 6));

-- Metadaten für räumliche Operatoren definieren
BEGIN
 insert into user_sdo_geom_metadata values ('DEMO_GAZ_DATA', 'SDO_LOCATION',
        SDO_DIM_ARRAY(SDO_DIM_ELEMENT('X', 0.08, 15.033816, .000005),
                      SDO_DIM_ELEMENT('Y', 47.274714, 55.056528, .000005)), null);
 insert into user_sdo_geom_metadata values ('DEMO_GAZ_RASTER', 'GEOMETRY',
        SDO_DIM_ARRAY(SDO_DIM_ELEMENT('X', 5.865003, 15.033816, .000005),
                      SDO_DIM_ELEMENT('Y', 47.274714, 55.056528, .000005)), null);
 insert into user_sdo_geom_metadata values ('DEMO_GAZ_RASTER_EXTENDED', 'GEOMETRY',
        SDO_DIM_ARRAY(SDO_DIM_ELEMENT('X', 5.865003, 15.033816, .000005),
                      SDO_DIM_ELEMENT('Y', 47.274714, 55.056528, .000005)), null);
 insert into user_sdo_geom_metadata values ('DEMO_GAZ_CLUS', 'GEOMETRY',
        SDO_DIM_ARRAY(SDO_DIM_ELEMENT('X', 5.865003, 15.033816, .000005),
                      SDO_DIM_ELEMENT('Y', 47.274714, 55.056528, .000005)), null);
END;
/

-- Index fuer die Tabellen erstellen
CREATE INDEX demo_gaz_data_sidx
        ON data_cities(SDO_LOCATION) indextype is MDSYS.SPATIAL_INDEX;
CREATE INDEX demo_gaz_raster_sidx
        ON demo_raster(GEOMETRY) indextype is MDSYS.SPATIAL_INDEX;


BEGIN
  DBMS_SDM_TRANSFORM.createRelMaterialisation(
          'def_materialisation_1','aggr_population','NUM_AC_REL');
  DBMS_SDM_TRANSFORM.insertNumACRelMaterialisation(
          'def_materialisation_1','demo_raster','id','geometry',
          'data_cities','sdo_location','SUM','population',NULL);
```

```
  DBMS_SDM_TRANSFORM.xformRelMaterialisation(
          'def_materialisation_1','demo_raster','id','demo_agglo_1');
END;
/
-- Dichte berechnen
CREATE TABLE demo_gaz_clus AS
      SELECT A.*, SDO_GEOM.SDO_AREA(A.geometry, 0.000005) as area,
             A.aggr_pop/SDO_GEOM.SDO_AREA(A.geometry, 0.000005) as density
      FROM demo_raster A;

CREATE INDEX demo_raster_sidx
       ON demo_raster(GEOMETRY) indextype is MDSYS.SPATIAL_INDEX;
CREATE INDEX demo_gaz_clus_sidx
       ON demo_gaz_clus(GEOMETRY) indextype is MDSYS.SPATIAL_INDEX;

 -- Ersetzung mit Funktionen aus dem SDM_TRANSFORM-Package
BEGIN
  DBMS_SDM_TRANSFORM.createRelMaterialisation(
          'def_materialisation_2','aggr_pop_neighbors','NUM_AC_REL');
  DBMS_SDM_TRANSFORM.insertNumACRelMaterialisation(
          'def_materialisation_2','demo_raster','id','geometry',
          'demo_agglo_1','geometry','SUM','aggr_population','sdo_num_res=8');
  DBMS_SDM_TRANSFORM.xformRelMaterialisation(
          'def_materialisation_2','demo_agglo_1','id','demo_agglo_2');
END;
/
-- Selektion der Daten, die der Definition genügen
CREATE VIEW demo_clus_selection AS
   SELECT * from demo_agglo_2 where aggr_pop_neighbors > 1000000 and density >
       100000;

BEGIN
  DBMS_SDM_TRANSFORM.create_SDM_Transform('demo_clustering', 'demo_clus_selection');
  DBMS_SDM_TRANSFORM.insert_SDM_Transform(
                              'demo_clustering', 'demo_clus_selection', 'geometry');
  DBMS_SDM_TRANSFORM.xform_SDM_Transform('demo_clustering');
END;
/

-----------------
-- DATA MINING
-- Konstruktion der Konfigurationstabelle und des Modells
CREATE TABLE test_clus_settings (setting_name  VARCHAR2(30),
                                 setting_value VARCHAR2(30));
BEGIN
     INSERT INTO test_clus_settings VALUES (DBMS_DATA_MINING.clus_num_clusters,8);
     INSERT INTO test_clus_settings VALUES (DBMS_DATA_MINING.kmns_iterations,5);
END;
/

BEGIN
        DBMS_DATA_MINING.CREATE_MODEL(
                model_name              => 'test_clus_1',
                mining_function         => DBMS_DATA_MINING.CLUSTERING,
                data_table_name         => 'demo_clustering',
                case_id_column_name     => 'sdm_id',
                settings_table_name     => 'test_clus_settings');
END;
/
-- Ausgabe der Centroide und Beschreibungen der Cluster
-- siehe Anhang B für Clustering
```

B Oracle Data Mining - Beispiele

Im Folgende wird der Quellcode der Beispiele zur Anwendung der Oracle Data Mining-Cartridge, die in Kapitel 4 vorgestellt und beschrieben wurden, aufgelistet.

B.1 Assoziationsanalyse

```
-- Daten für Assoziationsanalyse für Schauspieler (also wer mit wem?)
CREATE TABLE tmp_actor AS
 SELECT distinct at.movie, bt.person, bt.name
 FROM moviedb.part at join moviedb.person bt on (at.person=bt.person)
 WHERE at.mentioned = 'actor'
   AND at.person IN (SELECT person FROM moviedb.part
                     WHERE mentioned = 'actor' AND creditpos is not null
                     GROUP BY person
                     HAVING count(movie) > 20);

-- Anzahl der Schauspieler mit mind. 20 Filmen
CREATE TABLE asso_data_sample_5 AS
 SELECT movie as id, name as actor, 1 as present
 FROM tmp_actor
 WHERE movie IN (select movie from moviedb.movie
                 where year between 1950 and 2000 and type = 'cinema')
   AND movie IN (select movie from moviedb.country where country = 'USA')
   AND movie IN (select movie from tmp_actor
                 group by movie having count(movie) > 1)
 order by id, movie, name;

-- 1. Kardinalitäten von Movies, Schauspielern (, , )
select count(id) as anz_m, count(distinct id) as dist_anz_m,
       count(distinct actor) as Anz_Actor from asso_data_sample_5;

-- 2. Durchschnittliche Anzahl an Schauspieler pro Movie (avg = 3)
select to_char(avg(anz_actor), 999.99) avg_actor
 from (select count(actor) anz_actor
       from asso_data_sample_5
       group by id);

-- 3. Movie und Schauspieler als 'transactional data' darstellen
CREATE TABLE asso_prepared
NESTED TABLE actor_for_movie STORE AS DM_Nested_Categoricals_asso5
AS
(
  SELECT id,
      CAST(COLLECT(DM_Nested_Mumerical(
        SUBSTRB(actor, 1, 30), present))
      AS DM_Nested_Categoricals) actor_for_movie
  FROM asso_data_sample_5
  GROUP BY id
);
```

```
----------------
-- Konfigurationstabelle für den Apriori-Algorithmus
CREATE TABLE test_asso_settings (
       setting_name  VARCHAR2(30),
       setting_value VARCHAR2(30));
BEGIN
  INSERT INTO test_asso_settings VALUES (dbms_data_mining.asso_min_support,0.002);
  INSERT INTO test_asso_settings VALUES (dbms_data_mining.asso_min_confidence,0.5);
  INSERT INTO test_asso_settings VALUES (dbms_data_mining.asso_max_rule_length,6);
END;
/
-- Modell erstellen
BEGIN
       DBMS_DATA_MINING.CREATE_MODEL(
               model_name              => 'test_asso_1',
               mining_function         => DBMS_DATA_MINING.ASSOCIATION,
               data_table_name         => 'asso_prepared',
               case_id_column_name     => 'id',
               settings_table_name     => 'test_asso_settings');
END;
/
----------------
-- Frequent Itemsets ansehen
SELECT distinct item, support, number_of_items
 FROM (SELECT I.column_value AS item, F.support, F.number_of_items
         FROM TABLE(DBMS_DATA_MINING.GET_FREQUENT_ITEMSETS('test_asso_1')) F,
              TABLE(F.items) I
       ORDER BY number_of_items desc, support desc, column_value);
----------------
-- Assoziationsregeln
SELECT rule_id, rule_support as support, rule_confidence as confidence,
      B.attribute_name || B.conditional_operator || B.attribute_str_value as ante,
      C.attribute_name || C.conditional_operator || C.attribute_str_value as cons
 FROM TABLE(DBMS_DATA_MINING.GET_ASSOCIATION_RULES('test_asso_1')) A,
      TABLE(A.antecedent) B, TABLE(A.consequent) C;
```

B.2 Regressionsanalyse

```
-- Daten eingeschränkt: Filme zwischen 1990 und 2005
CREATE TABLE reg_data_sample_1 AS
 select rownum as id, year, anz_movie as target
   from ( select year, count(a.movie) as anz_movie from moviedb.movie a
           where year >= 1990 and year <= 2005 group by year)  f;

-- Normierung der Attribute
BEGIN
       DBMS_DATA_MINING_TRANSFORM.CREATE_NORM_LIN('conf_reg_sample_3_1');
       DBMS_DATA_MINING_TRANSFORM.INSERT_NORM_LIN_MINMAX(
               'conf_reg_sample_3_1',
               'reg_data_sample_1',
               DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST('id'));
       DBMS_DATA_MINING_TRANSFORM.XFORM_NORM_LIN(
               'conf_reg_sample_3_1', 'reg_data_sample_1', 'tmp_reg_prepared');
END;
/
-- Daten zur Konstruktion des Modells (build data) und zum Testen (test data)
CREATE TABLE reg_prepared_b as
```

```
   select id, year, target from tmp_reg_prepared natural join
        (select id, mod(rownum,3) as sel
         from tmp_reg_prepared where target is not null)
   where sel <> 1;

CREATE TABLE reg_prepared_t as
   select id, year, target from tmp_reg_prepared where target is not null
   minus
   select id, year, target from reg_prepared_b;
```

```
-- Konstruktion der Konfigurationstabelle und des Modells
CREATE TABLE test_reg_settings (setting_name  VARCHAR2(30),
                                setting_value VARCHAR2(30));

BEGIN
 INSERT INTO test_reg_settings VALUES (DBMS_DATA_MINING.svms_kernel_function,
                                       DBMS_DATA_MINING.svms_linear);
 DBMS_DATA_MINING.CREATE_MODEL(
          model_name              => 'test_reg_1',
          mining_function         => DBMS_DATA_MINING.REGRESSION,
          data_table_name         => 'reg_prepared_b',
          case_id_column_name     => 'id',
          target_column_name      => 'target',
          settings_table_name     => 'test_reg_settings');
END;
/
-- Display Details zum Modells
SELECT B.attribute_name, B.coefficient
FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_SVM('test_reg_1')) A,
     TABLE(A.attribute_set) B;

-------------------------
-- COMPUTE TEST METRICS
BEGIN
       DBMS_DATA_MINING.APPLY(
               model_name              => 'test_reg_1',
               data_table_name         => 'reg_prepared_t',
               case_id_column_name     => 'id',
               result_table_name       => 'test_reg_score');
END;
/
--------SCORING RESULTS -----
-- 1. Root Mean Square Error - Sqrt(Mean((x - x')^2))
SELECT SQRT(AVG((A.prediction - B.target) * (A.prediction - B.target))),2) rmse
FROM test_reg_score A NATURAL JOIN reg_prepared_t B;

-- 2. Mean Absolute Error - Mean(|(x - x')|)
SELECT AVG(ABS(a.prediction - B.target)) mae
FROM test_reg_score A NATURAL JOIN reg_prepared_t B

select (year*c.scale +c.shift) as YEAR, prediction, TARGET,
       abs(prediction - target) as DIFF,
       abs(prediction - target)*100 /target as ABWEICHUNG
from (select id, round((prediction*scale)+ shift,0) as prediction
        from test_reg_score, conf_reg_sample_3_1 where col ='TARGET') a
     natural join
     (select id, year ,(target*scale)+ shift as target
        from reg_prepared_t, conf_reg_sample_3_1 where col ='TARGET') b,
     conf_reg_sample_3_1 c
where c.col= 'YEAR' order by 1;
```

B.3 Clustering

```
-- Selektion der Daten
CREATE TABLE tmp_person as
 select person, min(year) as starting
 from moviedb.movie a join moviedb.part b on (a.movie=b.movie)
 where mentioned = 'actor'
 group by person;

CREATE TABLE clus_data_sample_9 AS
 select a.person as id, sex, height, nvl2(realname, '1', '0') as pseudo,
       nvl2(to_number(to_char(death_date, 'YYYY')), 0,
           (to_number(to_char(sysdate, 'YYYY')) -
            to_number(to_char(birth_date, 'YYYY'))) ) as age,
       b.starting - to_number(to_char(a.birth_date, 'YYYY')) as start_age
 from moviedb.person a join tmp_person b on (a.person=b.person)
 where height is not null and sex is not null and birth_date is not null
   and to_number(to_char(birth_date, 'YYYY')) > 1950;

-- Winsorizing der Daten  (Winsorizing liefert bessere Ergebnisse als Trimming)
BEGIN
 DBMS_DATA_MINING_TRANSFORM.CREATE_CLIP('conf_clus_sample_9_1');
 DBMS_DATA_MINING_TRANSFORM.INSERT_CLIP_WINSOR_TAIL(
               'conf_clus_sample_9_1', 'clus_data_sample_9', 0.025,
               DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST('id','pseudo'));
 DBMS_DATA_MINING_TRANSFORM.XFORM_CLIP(
               'conf_clus_sample_9_1', 'clus_data_sample_9', 'clus_temp_1');
END;
/
-- MinMax-Normierung
BEGIN
 DBMS_DATA_MINING_TRANSFORM.CREATE_NORM_LIN('conf_clus_sample_9_2');
 DBMS_DATA_MINING_TRANSFORM.INSERT_NORM_LIN_MINMAX(
               'conf_clus_sample_9_2', 'clus_temp_1',
               DBMS_DATA_MINING_TRANSFORM.COLUMN_LIST('id'));
 DBMS_DATA_MINING_TRANSFORM.XFORM_NORM_LIN(
               'conf_clus_sample_9_2', 'clus_temp_1', 'clus_prepared' );
END;
/
-- Konstruktion der Konfigurationstabelle und Parameter spezifizieren
CREATE TABLE test_clus_settings (setting_name  VARCHAR2(30),
                                 setting_value VARCHAR2(30));
BEGIN
 INSERT INTO test_clus_settings VALUES (DBMS_DATA_MINING.clus_num_clusters,8);
 INSERT INTO test_clus_settings VALUES (DBMS_DATA_MINING.algo_name,
                                        DBMS_DATA_MINING.algo_kmeans);
 INSERT INTO test_clus_settings VALUES (DBMS_DATA_MINING.kmns_iterations,5);
END;
/

-- Modell erstellen
BEGIN
    DBMS_DATA_MINING.CREATE_MODEL(
            model_name              => 'test_clus_1',
            mining_function         => DBMS_DATA_MINING.CLUSTERING,
            data_table_name         => 'clus_prepared',
            case_id_column_name     => 'id',
            settings_table_name     => 'test_clus_settings');
END;
/
```

```
-- Ausgabe der Regeln
select  A.rule.rule_id, A.rule.rule_support, A.rule.rule_confidence*100,
        B.attribute_name, B.conditional_operator, B.attribute_num_value, B.
            attribute_str_value,
        C.attribute_name, C.conditional_operator, C.attribute_num_value, C.
            attribute_str_value
from table(DBMS_DATA_MINING.GET_MODEL_DETAILS_KM ('test_clus_1')) A,
      table(A.rule.antecedent) B,
      table(A.rule.consequent) C,
      table(A.child) D
where D.ID IS NULL;
```

B.4 Klassifikation

```
-- Selektion der Daten
create table tmp_actor as
  select distinct at.movie, bt.name
  from moviedb.part at join moviedb.person bt on (at.person=bt.person)
  where at.mentioned = 'actor'
    and at.person IN (select person from moviedb.part
                      where mentioned = 'actor' and creditpos is not null
                      group by person
                      having count(movie) > 10);

-- Weitere Selektion und Diskretisierung des Attributs RATING
CREATE TABLE class_data_sample_3 AS
  select a.movie as id, b.name, 1 as name_present,
         CASE WHEN c.rating > 4 THEN CASE WHEN c.rating > 7 THEN 'hoch' else '
             mittel' END
                                ELSE 'niedrig' END as target
  from moviedb.movie a join tmp_actor b on (a.movie=b.movie)
                       join moviedb.rating c on (a.movie=c.movie)
  where type = 'cinema' and votes > 1000;

--
CREATE TABLE class_prepared
 NESTED TABLE actor_for_movie STORE AS DM_Nested_Numericals_actor
 AS
 (
  SELECT id, CAST(COLLECT(DM_Nested_Numerical(
                SUBSTRB(name, 1, 30), name_present))
            AS DM_Nested_Numericals) actor_for_movie, target
  FROM class_temp_1
  GROUP BY id, target
);

-- Daten zur Konstruktion des Modells (build data) und zum Testen (test data)
CREATE TABLE class_prepared_b
 NESTED TABLE actor_for_movie STORE AS DM_Nested_Numericals_actor_b
 AS
 (
   SELECT id, actor_for_movie, target
   FROM class_prepared natural join
           (select id, mod(rownum,25) as sel
            from class_prepared)
   WHERE sel <> 1
 );
```

```
CREATE TABLE class_prepared_t
 NESTED TABLE actor_for_movie STORE AS DM_Nested_Numericals_actor_t
 AS
 (
   SELECT id, actor_for_movie, target
   FROM class_prepared
   WHERE id NOT IN (
            SELECT id
            FROM class_prepared_b)
 );

-- Konstruktion der Konfigurationstabelle und des Modells
CREATE TABLE test_class_settings (setting_name  VARCHAR2(30),
                                  setting_value VARCHAR2(30));

BEGIN
  INSERT INTO test_class_settings VALUES (dbms_data_mining.algo_name,
                          dbms_data_mining.algo_support_vector_machines);
END;
/

BEGIN
       DBMS_DATA_MINING.CREATE_MODEL(
               model_name              => 'test_class_1',
               mining_function         => DBMS_DATA_MINING.CLASSIFICATION,
               data_table_name         => 'class_prepared_b',
               case_id_column_name     => 'id',
               target_column_name      => 'target',
               settings_table_name     => 'test_class_settings');
END;
/
-------------------------
-- COMPUTE TEST METRICS

-- Confusion-Matrix for TEST_DATEN
SELECT target AS actual_target_value,
      substr(PREDICTION(test_class_1 USING *),1,30) AS predicted_target_value,
      COUNT(*) AS Anzahl
FROM class_prepared_t
GROUP BY target, PREDICTION(test_class_1 USING *)
ORDER BY 1;

SELECT round(SUM(DECODE(target, PREDICTION(test_class_1 USING *), 1, 0))
             /count(*), 4)*100 || '%' as Accuracy
FROM class_prepared_t;


-- Confusion-Matrix for COMPLETE_DATEN
SELECT target AS actual_target_value,
      substr(PREDICTION(test_class_1 USING *),1,30) AS predicted_target_value,
      COUNT(*) AS Anzahl
FROM class_prepared
GROUP BY target, PREDICTION(test_class_1 USING *)
ORDER BY 1;

SELECT round(SUM(DECODE(target, PREDICTION(test_class_1 USING *), 1, 0))
             /count(*), 4)*100 || '%' as Accuracy
FROM class_prepared;
```

```
-- Neue Daten werden mit dem Klassifikator klassifiziert (TESTDATEN mit
    Vergleichswert)
SELECT id,
       b.rating,
       a.target AS actual_target_value,
       substr(PREDICTION(test_class_1 USING a.*),1,30) AS predicted_target_value
FROM class_prepared_t a join moviedb.rating b on (a.id=b.movie)
WHERE rownum < 20;
```

Literaturverzeichnis

[AMJ99] K. H. Ankerst M., Breunig M., S. J.: OPTICS: Ordering Points To Identify the Clustering Structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, ACM Press, 1999, 49–60.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo: Fast discovery of association rules. , 307–328.

[Atk06] ATKIS-Amtliches Topographisch-Kartographisches Informationssystem. 2006.

[BC00] K. P. Bennett, C. Campbell: Support Vector Machines: Hype or Hallelujah? *SIGKDD Explorations 2:2 (2000)*, 1–13.

[BM05] M. C. B.L. Milenova, J.S. Yarmus: SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines. In *Proc. of the 31st VLDB Conference, Trondheim, Norway, 2005*, 2005.

[Cha91] E. Charniak: Bayesian Networks without Tears. *AI Magazine (1991)*.

[CP97] M. A. Carreira-Perpinan: A Review of Dimension Reduction Techniques. Technical Report CS-96-09, Dept. of Computer Science, University of Sheffield, 1997.

[DKS95] J. Dougherty, R. Kohavi, M. Sahami: Supervised and Unsupervised Discretization of Continuous Features. In *International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, 1995, 194–202.

[DL99] H. S. D.D. Lee: Learning the parts of objects by nonnegative matrix factorization. *Nature 401 (1999)*, 788–791.

[DLR77] A. P. Dempster, N. M. Laird, D. B. Rubin: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39:1 (1977)*, 1–38.

[Dub87] R. C. Dubes: How many clusters are best? an experiment. *Pattern Recogn. 20:6 (1987)*, 645–663.

[EFKS98] M. Ester, A. Frommelt, H.-P. Kriegel, J. Sander: Algorithms for Characterization and Trend Detection in Spatial Databases. In *KDD*, 1998, 44–50.

[Ege91] M. J. Egenhofer: Reasoning about Binary Topological Relations. In *SSD '91: Proceedings of the Second International Symposium on Advances in Spatial Databases*, Springer-Verlag, London, UK, 1991, 143–160.

[EKS97] M. Ester, H.-P. Kriegel, J. Sander: Spatial Data Mining: A Database Approach. In M. Scholl, A. Voisard (eds.), *Fifth Symposium on Large Spatial Databases (SSD'97)*, volume 1262, Springer, Berlin, Germany, 1997, 48–66.

[EN00] R. Elmasri, S. B. Navathe: *Fundamentals of Database Systems (Third Edition)*, 3rd edition. World Student Series, Addison-Wesley, Reading, MA, 2000.

[FAE06] Lorentzkurve – http://www.faes.de/Basis/Basis-Lexikon/Basis-Lexikon-Lorenz-Kurve/basis-lexikon-lorenz-kurve.html. 2006.

[FI93] Fayyad, Irani: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. 1993, 1022–1027.

[Fod02] I. Fodor. A Survey of Dimension Reduction Techniques. 2002.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth: From Data Mining to Knowledge Discovery: An Overview. In *Advances in Knowledge Discovery and Data Mining*, 1996, 1–34.

[FPSSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy: *Advances in Knowledge Discovery and Data Mining*. The MIT Press, Cambridge, MA, 1996.

[Fri89] J. Friedman: Regularized discriminant analysis. *Journal of the American Statistical Association 84 (1989)*, 165–175.

[FSM92] W. J. Frawley, P. G. Shapiro, C. J. Matheus: Knowledge discovery in databases - an overview. *Ai Magazine 13 (1992)*, 57–70.

[Gaz06] Gazetteer-Projekt – http://world-gazetteer.com/. 2006.

[Haw80] D. Hawkins: *Identification of Outliers*. Chapman and Hall, London, 1980.

[Hec95] D. Heckerman. A tutorial on learning with bayesian networks. 1995.

[HF95] J. Han, Y. Fu: Discovery of Multiple-Level Association Rules from Large Databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*, 1995, 420–431.

[HK00] J. Han, M. Kamber: *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2000.

[HL02] G. J. Hwang, F. Li: A Dynamic Method for Discretization of Continuous Attributes. In *Intelligent Data Engineering and Automated Learning - IDEAL 2002, Third International Conference, Manchester, UK, August 12-14, Proceedings*, volume 2412 of *Lecture Notes in Computer Science*, Springer, 2002, 506–511.

[HPX06] Y. Huang, J. Pei, H. Xiong: Mining Co-Location Patterns with Rare Events from Spatial Data Sets. *Geoinformatica 10:3 (2006)*, 239–260.

[HSC91] R. Hanson, J. Stutz, P. Cheeseman: Bayesian classification theory. Technical report, 1991.

[JMF99] A. K. Jain, M. N. Murty, P. J. Flynn: Data clustering: a review. *ACM Computing Surveys 31:3 (1999)*, 264–323.

[Joa] K. M. S. W. T. Joachims: *Maschinelles Lernen und Data Mining*. G. Görz, J. Schneeberger und C.-R. Rollinger im Oldenbourg Verlag.

[KAH96] K. Koperski, J. Adhikary, J. Han: Knowledge Discovery in Spatial Databases: Progress and Challenges. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, Montreal, Canada, 1996, 55–70.

[KH95] K. Koperski, J. Han: Discovery of Spatial Association Rules in Geographic Information Databases. In M. J. Egenhofer, J. R. Herring (eds.), *Proc. 4th Int. Symp. Advances in Spatial Databases, SSD*, volume 951, Springer-Verlag, 1995, 47–66.

[KHN99] G. Karypis, E.-H. S. Han, V. K. NEWS: Chameleon: Hierarchical Clustering Using Dynamic Modeling. *Computer 32:8 (1999)*, 68–75.

[KHS] K. Koperski, J. Han, N. Stefanovic. An Efficient Two-Step Method for Classification of Spatial Data.

[Kie06] D. J.-U. Kietz: *Data Mining zur Wissensgewinnung aus Datenbanken*. Univ. Zürich, SoSe 2006.

[KK96] D. A. Keim, H.-P. Kriegel: Visualization Techniques for Mining Large Databases: A Comparison. *Transactions on Knowledge and Data Engineering, Special Issue on Data Mining 8:6 (1996)*, 923–938.

[KN97] E. M. Knorr, R. T. Ng: A Unified Notion of Outliers: Properties and Computation. In *Knowledge Discovery and Data Mining*, 1997, 219–222.

[KR87] L. Kaufman, P. J. Rousseeuw: Clustering by means of medoids. , 405–416.

[KR90] L. Kaufman, P. J. Rousseeuw: Finding Groups in Data, An Itroduction to Cluster Analysis.

[Len02] M. Lenzerini: Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM Press, 2002, 233–246.

[Lew98] D. D. Lewis: Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, Springer Verlag, Heidelberg, DE, 1998.

[Lip04] U. Lipeck: *Datenbanksysteme IIa - Kapitel über das Zusammenführen von Datenbanken (ER-Diagramme)*. Universität. Hannover, 2004.

[Los06] D. Loss: *Data Mining: Klassifikations- und Clusteringverfahren.* Westfälische Wilhelms-Universität Münster, 2006.

[Mac67] J. MacQueen: Some Methods for Classification and Analysis of Multivariate Observation. , 281–297.

[Mar05] J. B. Marshall: *STAT 3005 – Statistical Methods*. Virginia Tech, 2005.

[MC02] B. L. Milenova, M. M. Campos: O-Cluster: Scalable Clustering of Large High Dimensional Data Sets. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, IEEE Computer Society, Washington, DC, USA, 2002, 290.

[ME96] J. S. X. X. Martin Ester, Hans-Peter Kriegel: A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996, 226–231.

[Mit97] T. M. Mitchell: *Machine learning*. New York, NY: McGraw-Hill, SoSe 1997.

[Mor01] Y. Morimoto: Mining frequent neighboring class sets in spatial databases. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press, New York, NY, USA, 2001, 353–358.

[NH94] R. T. Ng, J. Han: Efficient and Effective Clustering Methods for Spatial Data Mining. In J. Bocca, M. Jarke, C. Zaniolo (eds.), *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1994, 144–155.

[NH02] R. T. Ng, J. Han: CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering 14:5 (2002)*, 1003–1016.

[NMV92] L. B. J. D. N. Matic, I. Guyon, V. Vapnik: Computer Aided Cleaning of Large Databases for Character Recognition. In *Proceedings of the 11th Int. Conf. on Pattern Recognition*, IEEE Computer Society Press, 1992, 330–333.

[Oraa] Oracle: Oracle Database 10g – Empowering Applications with Spatial Analysis and Mining. Technical report, Februar.

[ORAb] Oracle® Spatial User's Guide and Reference.

[ORA06a] Oracle Forum. 2006.

[Ora06b] Application Developer's Guide, 10g Release 2. 2006.

[Ora06c] Oracle Data Mining Reference. 2006.

[Ora06d] Oracle Data Mining: Conecpts, 10g Release 2. 2006.

[Ora06e] Oracle Data Mining: Conecpts - Requirements. 2006.

[Ora06f] Oracle® Database PL/SQL Packages and Types Reference 10g Release 2. 2006.

[Ora06g] Spatial Analysis and Mining. 2006.

[Ora06h] Oracle® Database SQL Reference 10g Release 2. 2006.

[PT98] P. Perner, S. Trautzsch: Multi-interval Discretization Methods for Decision Tree Learning. In *SSPR/SPR*, Lecture Notes in Computer Science, Springer, 1998, 475–482.

[Qui93] J. R. Quinlan: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Ris78] J. Rissanen: Modeling by the shortest data description. *Automatica 14 (1978)*, 465–471.

[ROC06] ROC (Receiver Operating Characteristics) – http://www.anaesthetist.com/mnm/-stats/roc/Findex.htm. 2006.

[RT04] S. Rinzivillo, F. Turini: Classification in geographical information systems. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Springer-Verlag New York, Inc., New York, NY, USA, 2004, 374–385.

[RT05] S. Rinzivillo, F. Turini: Extracting spatial association rules from spatial transactions. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, ACM Press, New York, NY, USA, 2005, 79–86.

[SC03] S. Shekhar, S. Chawla: *Spatial Databases: A Tour*. Prentice Hall, Upper Saddle River, 2003.

[Sch05] D. M. Schubert: *Knowledge Discovery and Data Mining*. Univ. Heidelberg, SoSe 2005.

[Sch06] S. Schosser: *Praktikum: Data Warehousing und Data Mining.* Universität Karlsruhe, 2006.

[SEKX98] J. Sander, M. Ester, H.-P. Kriegel, X. Xu: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Min. Knowl. Discov. 2:2 (1998)*, 169–194.

[SH00] M. Skubacz, J. Hollmen: Quantization of Continuous Input Variables for Binary Classification. In *Intelligent Data Engineering and Automated Learning - IDEAL 2000*, 2000, 42–47.

[SS98] A. Smola, B. Schoelkopf. A tutorial on support vector regression. 1998.

[SSG01] J.-W. H. S. C. Shashi Shekhar, Yan Huang, S. Gopal: Categorization of Spatial Data Mining Techniques. *Scientific Data Mining, working chapter (2001)*.

[Säu00] F. Säuberlich: *KDD und Data Mining als Hilfsmittel zur Entscheidungsunterstützung*. Peter Lang Verlag, Frankfurt am Main, 2000.

[Tie03] M. Tiedge. Entwicklung und Implementierung einer topologischen Erweiterung für objektbasierte räumliche Datenbanken. Institut für Informationssysteme, Universität Hannover, Hannover, 2003.

[Tob79] W. R. Tobler: CELLULAR GEOGRAPHY. In *Philosophy in Geography*, D. Reidel Publishing Company, Dordrecht, Holland, 1979, 379–386.

[TZ96] M. L. T. Zhang, R. Ramakrishnan: BIRCH: An efficient data clustering method for very large databases. In *In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada*, 1996, 103–114.

[Vap95] V. N. Vapnik: *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[VB05a] L. O. A. Vania Bogorny, Paulo Martins Engel: A Reuse-based Spatial Data Preparation Framework for Data Mining. *IEEE Computer Graphics 24:5 (2005)*, 16–19.

[VB05b] L. O. A. Vania Bogorny, Paulo Martins Engel: Spatial Data Preparation for Knowledge Discovery. *IEEE Computer Graphics 24:5 (2005)*, 16–19.

[VB06a] A. T. Vania Bogorny: Extending the WEKA Data Mining Toolkit to Support Geographic Data Preprocessing. *Technical Report RP354 (2006)*.

[VB06b] P. M. E. L. O. A. Vania Bogorny, Andrey Tietbohl Palma: WEKA-GDPM – Integrating Classical Data Mining Toolkit to Geographic Information Systems. In *In: SBBD Workshop on Data Mining Algorithms and Applications(WAAMD'06)*, Florianopolis, Brazil, 2006, 9–16.

[War04] M. Ward: Finding Needles in Large-Scale Multivariate Data Haystacks. *IEEE Computer Graphics 24:5 (2004)*, 16–19.

[WEK] WEKA 3 Data Mining Software in Java.

[WIK06] Wikipedia - The Free Encyclopedia. 2006.

[WYM97] W. Wang, J. Yang, R. R. Muntz: STING: A Statistical Information Grid Approach to Spatial Data Mining. In *Twenty-Third International Conference on Very Large Data Bases*, Morgan Kaufmann, Athens, Greece, 1997, 186–195.

[YLG+05] W. Yang, Y. Luo, P. Guo, H. Tao, B. He: A Model for Classification of Topological Relationships Between Two Spatial Objects. In *FSKD (2)*, 2005, 723–726.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit und die zugehörige Implementierung selbständig verfasst und dabei nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Hannover, 30. März 2007

Markus Spehling